全国高等职业教育计算机类规划教材·工作过程系统化教程系列
2008 年国家精品课程配套教材

Java 项目实战精编

陈显刚　李　季　主　编
张　静　孙凌玲　副主编
姜惠民　主　审

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书是为了满足新世纪高等职业学校教学的需要而编写的教材。本书较全面地介绍了 Java 基本设计和应用技术，内容包括面向对象的技术、Java Swing 技术、IO 技术、线程技术和网络技术及 Java 项目开发的过程等。

本书以奠定 Java 编程思维模式、培养 Java 项目开发能力为目标，注重 Java 项目开发技术的实用，通过项目介绍 Java 知识体系，由浅入深、循序渐进，符合认知规律及职业发展规划，并配有项目案例库。

本书可作为高等职业学院计算机专业专科及本科学生的教材，也可供与计算机相关专业的技术人员使用。

图书在版编目（CIP）数据

Java 项目实战精编 / 陈显刚，李季主编. —北京：电子工业出版社，2009.7
全国高等职业教育计算机类规划教材·工作过程系统化教程系列
ISBN 978-7-121-08966-4

Ⅰ. J… Ⅱ. ①陈…②李… Ⅲ. JAVA 语言－程序设计－高等学校：技术学校－教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2009）第 086181 号

策划编辑：程超群
责任编辑：王凌燕
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：14. 5　字数：368 千字
印　　次：2009 年 7 月第 1 次印刷
印　　数：4 000 册　　定价：23.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

Java 语言是一种新型的网络编程语言，其卓越的特性为无数开发人员所推崇，目前越来越多的应用开发采用了基于 Java 技术的解决方案。Java 是一种简单的，面向对象的，分布式的，解释型的，健壮安全的，结构中立的，可移植的，性能优异、多线程的动态语言。作为一种真正面向对象的编程语言，它提升了应用程序的编程概念和开发思路；作为理想的面向对象的程序设计语言，Java 以自身的简单性和强大功能成为 Internet 编程和跨平台开发中最常用的开发语言。

Java 语言具有面向对象、与平台无关、安全、稳定和多线程等优良特性，是目前软件设计中极为强大的编程语言。Java 语言不仅可以用来开发大型的应用程序，而且特别适合 Internet 的应用开发。尤其是 Java Swing 推出之后，不仅使 Java 的功能更加强大，而且使 Java 具备了“处处可用”的特点，Java 已成为网络时代最重要的语言之一。

本书是工作过程导向系统化课程教材，是行动体系课程开发的成果。以培养能力为主线，按工作过程中不同工作任务的相关性来实现知识和实践技能的整合。按易学、易懂、易掌握的原则，结合 Java 技术，由浅入深，循序渐进地，通过项目介绍 Java 知识体系。

本书聘请启明信息科技股份有限公司高级软件工程师杨平参编。作为 ERP 项目组项目经理，他具有丰富的实践开发经验。他对本书的编写模式、项目设计思想、编码规范等方面给予指导，并根据企业常用的实际知识和技能，设计全书项目，以案例引领知识点，拓宽程序设计思路，通过实训项目提高实践技能。

全书共分 5 章。第 1 章通过掷骰子游戏项目，阐述面向对象的技术；第 2 章通过简单计算器项目，简述 Java Swing 技术；第 3 章通过聊天室项目，简述 IO 技术、线程技术和网络技术等；第 4 章通过二十一点游戏项目，加强 Java Swing 技术和分析问题的能力；第 5 章通过学生信息管理系统综合项目，进一步强化项目开发的能力，同时掌握软件开发过程。通过 5 个项目使学生理解并掌握利用 Java 技术解决实际问题的能力，而不是就 Java 技术而学 Java 技术。

本书的最大特点是通过项目对 Java 的知识点进行精心编排。项目设计顺序符合认知规律及职业规划发展规律，通过对项目的学习，加深读者对所学知识的理解和提升。通过对应的实训项目训练，提高分析问题和解决问题的能力。

本书由陈显刚、李季主编，张静、孙凌玲任副主编；参与本书编写的还有张雨、孙佳帝、金鑫、许春艳、乔丹、杨平；由姜惠民主审。

由于计算机技术发展十分迅速以及作者学识水平所限，加之时间仓促，书中的疏漏和错误在所难免，敬请广大读者不吝批评指正。

编　者

2009 年 4 月

目　录

第 1 章　掷　骰　子

1.1　项目目标

丢下两个骰子，若数值的总值为 7 点，则赢；否则输。

1.2　项目分析

项目的操作过程如下：3

（1）定义一个表示骰子的类，能掷出数字及获取当前掷出的数。

（2）定义另一个类，在此类中建立骰子类的两个对象，当掷出时判断两个骰子的数值的总值是否为 7，来确定是否成功。

（3）定义第三个类，对以上操作进行测试。

1.3　代码思路及实现

1.3.1　代码思路

1．定义Die类

Die 类表示一个骰子有一个 faceValue 属性，为整型。在 Die 类中有 roll()方法和 getFaceValue()方法。roll()方法功能使 faceValue 属性设置为 1～6 中的一个随机值，没有返回值。getFaceValue()方法功能为取出 faceValue 值。

在本类中使用到了 Math.random()方法，此方法可以实现取随机数的功能，返回值为 double 类型，值的范围为 0.0～1.0。

2．定义DieGame类

提示：DieGame 类有 die1、die2 两个属性，类型分别为 Die 类类型，有一个 play()方法。play()方法返回一个布尔类型，true 表示丢下两个骰子数值的总值为 7 点，否则为 false。

3．编写一个测试类DieTest，对上面定义的类进行测试

main()方法中产生 DieGame 对象，执行 play()方法后显示出输赢。

1.3.2 代码实现

1．Die类

```
import java.util.*;
public class Die {
    private int faceValue;
    public void roll()
    {
        Random random=new Random();

        this.faceValue=Math.abs(random.nextInt()) %6+1;//取 1～6 间任意整数
    }
    public    int getFaceValue()
    {
        return this.faceValue;
    }
}
```

2．DieGame类

```
public class DiceGame {

    public Die die1=new Die();
    public Die die2=new Die();

    public boolean play()                                    //掷骰子，两粒骰子数相加得 7 为 true，否则为 false
    {
        boolean flag=false;

        die1.roll();
        die2.roll();

        System.out.println(die1.getFaceValue());
        System.out.println(die2.getFaceValue());     //输出本次掷得的骰子数

        int num=die1.getFaceValue()+die2.getFaceValue();
        if(num==7)
        {
            flag=true;
        }
        else
        {
            flag=false;
        }
        return flag;
```

```
    }
}
```

3．测试类DieTest

```
public class DieTest {
    public static void main(String[] args) {
        DiceGame dg=new DiceGame();
        if(dg.play())
        {
            System.out.println("您赢了!");
        }
        else
        {
            System.out.println("您输了,请下次努力!");
        }
    }
}
```

1.4 运行与发布

1.4.1 运行

将 Die.java、DieGame.java 和 DieTest.java 3 个文件保存到一个文件夹中，如 e:\Die。在使用 javac 命令进行编译之前，应使用如下命令设置类路径：

```
e:\Die >set classpath= e:\Die
```

然后利用 javac 命令对文件进行编译，使用如下命令：

```
Javac DieTest.java
```

之后，使用 java 执行程序：

```
Java DieTest
```

程序即运行。

1.4.2 发布

使用 jar.exe 将应用程序打包，把应用程序中涉及的类和图片压缩成一个 jar 文件，这样就可以发布程序了。

步骤如下：

（1）编写清单文件，名为 MANIFEST.MF，保存到 e:\Die 文件夹下。其代码如下：

```
Manifest-Version: 1.0
Created-By: 1.5.0_02(Sun Microsystems Inc.)
Main-Class: Die
```

（2）使用如下命令生成 jar 文件：

```
jar cfm Die.jar MANIFEST.MF *.class
```

其中，c 表示要生成一个新的 jar 文件；f 表示要生成的 jar 文件的名字；m 表示清单文件的名字。

（3）为解决解压软件与.jar 文件的关联问题，在发布软件时还应该再编写一个 Die.bat 文件。其中只有如下一条命令：

```
javaw –jar Die.jar
```

以后就可以通过双击 Die.bat 来运行程序了。

1.5 本项目实现中常见问题

可以使用 java.util.Random 类来实现取随机数。取 1～6 中的任意整数时的表达式为：

```
Math.abs(random.nextInt()) %6+1
```

另外可以使用 Math.random()来实现取随机数的操作。

Math.random()的取值范围是 0～1 之间的任意小数，包括 0 但不包括 1，所以在取 1～6 中的任意整数时表达式要写成：

```
(int)(Math.random()*6)+1
```

1.6 项目技术支持

1.6.1 面向对象的基本概念

面向对象是一种新兴的程序设计方法，或者说是一种新的程序设计规范（paradigm），其基本思想是使用对象、类、继承、封装、消息等基本概念来进行程序设计。从现实世界中客观存在的事物（即对象）出发来构造软件系统，并且在系统构造中尽可能运用人类的自然思维方式。开发一个软件是为了解决某些问题，这些问题所涉及的业务范围称做该软件的问题域。其应用领域不仅仅是软件，还有计算机体系结构和人工智能等。

1．对象的基本概念

对象是系统中用来描述客观事物的一个实体，它是构成系统的一个基本单位。一个对象由一组属性和对这组属性进行操作的一组服务组成。从更抽象的角度来说，对象是问题域或实现域中某些事物的一个抽象，它反映该事物在系统中需要保存的信息和发挥的作用；它是一组属性和有权对这些属性进行操作的一组服务的封装体。客观世界是由对象和对象之间的联系组成的。

主动对象是一组属性和一组服务的封装体，其中至少有一个服务不需要接收消息就能主动执行（称做主动服务）。

2．类的基本概念

把众多的事物归纳、划分成一些类是人类在认识客观世界时经常采用的思维方法。分类

的原则是抽象。类是具有相同属性和服务的一组对象的集合，它为属于该类的所有对象提供了统一的抽象描述，其内部包括属性和服务两个主要部分。在面向对象的编程语言中，类是一个独立的程序单位，它应该有一个类名并包括属性说明和服务说明两个主要部分。类与对象的关系就如模具和铸件的关系，类的实例化结果就是对象，而对一类对象的抽象就是类。

3．消息

消息就是向对象发出的服务请求，它应该包含下述信息：提供服务的对象标识、服务标识、输入信息和回答信息。服务通常被称为方法或函数。

1.6.2 面向对象的基本特征

1．封装性

封装性就是把对象的属性和服务结合成一个独立的相同单位，并尽可能隐蔽对象的内部细节。它包含以下两个含义：

（1）把对象的全部属性和全部服务结合在一起，形成一个不可分割的独立单位（即对象）。

（2）信息隐蔽，即尽可能隐蔽对象的内部细节，对外形成一个边界（或者说形成一道屏障），只保留有限的对外接口使之与外部发生联系。

封装的原则在软件上的反映是：要求使对象以外的部分不能随意存取对象的内部数据（属性），从而有效地避免了外部错误对它的“交叉感染”，使软件错误能够局部化，大大减少查错和排错的难度。

2．继承性

特殊类的对象拥有其一般类的全部属性与服务，称做特殊类对一般类的继承。例如，轮船、客轮，人、大人。一个类可以是多个一般类的特殊类，它从多个一般类中继承了属性与服务，这称为多继承。例如，客轮是轮船和客运工具的特殊类。在 java 语言中，通常称一般类为父类（superclass，超类），特殊类为子类（subclass）。

3．多态性

对象的多态性是指在一般类中定义的属性或服务被特殊类继承之后，可以具有不同的数据类型或表现出不同的行为。这使得同一个属性或服务在一般类及其各个特殊类中具有不同的语义。例如，“几何图形”的“绘图”方法，“椭圆”和“多边形”都是“几何图”的子类，其“绘图”方法功能不同。

1.6.3 类

类是 Java 中的一种重要的复合数据类型，是组成 Java 程序的基本要素。它封装了一类对象的状态和方法，是这一类对象的原形。一个类的实现包括两个部分：类声明和类体。

1．类声明

```
[public][abstract|final] class className [extends superclassName] [implements interfaceNameList]
{…}
```

其中，修饰符 public、abstract、final 说明了类的属性，className 为类名，superclassName 为类的父类的名字，interfaceNameList 为类所实现的接口列表。

2．类体

类体定义如下：

```
class className
{[public | protected | private ] [static]
   [final] [transient] [volatile] type
   variableName;                    //成员变量
   [public | protected | private ] [static]
   [final | abstract] [native] [synchronized]
   returnType methodName([paramList]) [throws exceptionList]
   {statements}                     //成员方法
}
```

3．成员变量

成员变量的声明方式如下：

```
[public | protected | private ] [static] [final] [transient] [volatile] type
variableName;                //成员变量
```

其中，
static：静态变量（类变量），相对于实例变量。
final：常量。
transient：暂时性变量，用于对象存档。
volatile：贡献变量，用于并发线程的共享。

4．成员方法

方法的实现包括两部分内容：方法声明和方法体。

```
[public | protected | private ] [static]
[final| abstract] [native] [synchronized]
returnType methodName([paramList])
[throws exceptionList]       //方法声明
{statements}                 //方法体
```

方法声明中的限定词的含义如下。
static：类方法，可通过类名直接调用。
abstract：抽象方法，没有方法体。
final：方法不能被重写。
native：集成其他语言的代码。
synchronized：控制多个并发线程的访问。

1.6.4 对象

类实例化可生成对象，对象通过消息传递来进行交互。消息传递即激活指定的某个对象的方法以改变其状态或让它产生一定的行为。一个对象的生命周期包括三个阶段：生成、使用和消除。类与对象的关系如图 1-1 所示。

1．对象的生成

对象的生成包括声明、实例化和初始化。

格式为：

```
type objectName=new type([paramlist]);
```

（1）声明：type objectName。声明并不为对象分配内存空间，而只是分配一个引用空间；对象的引用类似于指针，是 32 位的地址空间，它的值指向一个中间的数据结构，它存储有关数据类型的信息以及当前对象所在的堆的地址，而对于对象所在的实际的内存地址是不可操作的，这就保证了安全性。

（2）实例化：运算符 new 为对象分配内存空间，它调用对象的构造方法，返回引用；一个类的不同对象分别占据不同的内存空间。

（3）生成：执行构造方法，进行初始化；根据参数不同调用相应的构造方法。

```
Student s1=new Student();
```

上面语句按照 Student 类生成一个 s1 对象，或者用下面两条语句。

```
Student s1;
s1=new Student();
```

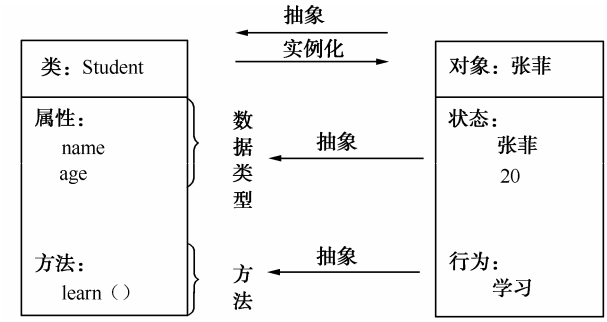

图 1-1 类与对象关系

2．对象的使用

通过运算符“.”可以实现对变量的访问和方法的调用。变量和方法可以通过设定访问权限来限制其他对象对它的访问。

（1）调用对象的变量。

格式：对象名.变量

对象名是一个已生成的对象，也可以是能生成对象的表达式。

例如：s.name="张三";

（2）调用对象的方法。

格式：对象名.方法名([paramlist]);

例如：s.show( );

3．对象的清除

当不存在对一个对象的引用时，该对象成为一个无用对象。Java 的垃圾收集器自动扫描对象的动态内存区，把没有引用的对象作为垃圾收集起来并释放。

```
System.gc( );
```

当系统内存用尽或调用 System.gc( )要求垃圾回收时，垃圾回收线程与系统同步运行。

1.6.5 方法

一个类中可以有很多方法，这些方法用来实现类某些功能或动作。

1．局部变量与类的成员变量同名

方法体是对方法的实现，它包括局部变量的声明以及所有合法的 Java 指令。方法体中声明的局部变量的作用域在该方法内部。若局部变量与类的成员变量同名，则类的成员变量被隐藏。

【例 1-1】　本例说明了局部变量 z 和类成员变量 z 的作用域是不同的。

```
class Variable{
    int x=0,y=0,z=0;                    //类的成员变量
    void init(int x,int y) {
    this.x=x;
    this.y=y;
    int z=5;                            //局部变量
    System.out.println("** in init**");
    System.out.println("x="+x+" y="+y+" z="+z);
  }
}
public class VariableTest{
    public static void main(String args[]){
    Variable v=new Variable();
    System.out.println("**before init**");
    System.out.println("x="+v.x+" y="+ v.y+" z="+v.z);
    v.init(20,30);
    System.out.println("**after init**");
    System.out.println("x="+v.x+ " y="+ v.y+" z="+v.z);
  }
}
```

运行结果为：

```
**before init**
x=0 y=0 z=0
** in init **
```

```
x=20 y=30 z=5
**after init**
x=20 y=30 z=0
```

上例中用到了 this，这是因为 init()方法的参数名与类的成员变量 x,y 的名字相同，而参数名会隐藏成员变量，所以在方法中，为了区别参数和类的成员变量，必须使用 this。

this：用在一个方法中引用当前对象，它的值是调用该方法的对象。返回值须与返回类型一致，或者完全相同，或是其子类。当返回类型是接口时，返回值必须实现该接口。

2．方法中的参数传递

对于简单数据类型来说，java 实现的是值传递，方法接收参数的值，但不能改变这些参数的值。如果要改变参数的值，则用引用数据类型，因为引用数据类型传递给方法的是数据在内存中的地址，方法中对数据的操作可以改变数据的值。

（1）按值传递方式。

【例 1-2】　按值传递实例演示。

```
class Test{
   static show add(int a){
        a=a+10;
   }
  public static void main(String args[]){
       int a=10;
       System.out.println("调用方法前 a 的值为"+a);
       Test.add(a);
       System.out.println("调用方法后 a 的值为"+a);
     }
}
```

运行结果为：

调用方法前 a 的值为 10

调用方法后 a 的值为 10

Press any key to continue...

（2）按地址传递方式。

【例 1-3】　按地址传递实例演示。

```
class Student{
   String name;
   void change(Student op){
      name=op.name;
  }
}
class TestStudent{
  public static void main(String args[]){
        Student s1=new Student();
        s1.name="Jack";
        Student s2=new Student();
```

```
        s2.name="Tom";
        System.out.println("打印前的 s1.name 的值是"+s1.name);
        s1.change(s2);
        System.out.println("打印后的 s1.name 的值是"+s1.name);
    }
}
```

运行结果为：

打印前的 s1.name 的值是 Jack

打印后的 s1.name 的值是 Tom

Press any key to continue...

3．方法重载

方法重载是指多个方法享有相同的名字，但是这些方法的参数必须不同，或者是参数的个数不同，或者是参数类型不同。返回类型不能用来区分重载的方法。

参数类型的区分度一定要足够，如不能是同一简单类型的参数，如 int 与 long。

【例 1-4】　方法重载实例演示。

```
class Abs{
    int abs(int a){
        if(a>0)
            return a;
        else
            return −a;
    }
    long abs(long a){
        if(a>0)
            return a;
        else
            return −a;
    }
    double abs(double a){
        if(a>0)
            return a;
        else
            return −a;
    }
}
class AbsTest{
public static void main(String arg[]) {
        Abs ob=new Abs();
        int result1;
        long result2;
        double result3;
        result1=ob.abs(5);
        result2=ob.abs(23456789);
```

```
        result3=ob.abs(2.0);
        System.out.println("result of ob.abs(−5):"+result1);
        System.out.println("result of ob.abs(−123456789):"+result2);
        System.out.println("result of ob.abs(−2.0):"+result3);
    }
    }
```

运行结果为：

```
result of ob.abs(−5):5
result of ob.abs(−123456789):23456789
result of ob.abs(−2.0):2.0
Press any key to continue...
```

从此例中可以看出，abs 方法是如何区分的，它主要是通过参数的类型和返回值来区分的。

4．构造方法

（1）构造方法是给类中的成员变量赋初值，没有返回值。

（2）构造方法与它所在的类同名。

（3）构造方法不能被程序显示调用。

（4）构造方法可以有零个或多个自变量。

（5）构造方法可以在类中由编程者定义，如果编程者没有定义，系统将自动生成一个构造函数。

构造方法可以通过重载实现不同的初始化方法。

【例 1-5】　构造方法实例说明。

```
class Point{
   int x;
   int y;
   Point(){
      x=0;
      y=0;
   }
   Point(int x, int y){
      this.x=x;
      this.y=y;
   }
   Point(Point op){
      x=op.x;
      y=op.y;
   }
}
class TestPoint{
public static void main(String args[]){
      Point p1=new Point();
      System.out.println("p1.x="+p1.x);
      System.out.println("p1.y="+p1.y);
```

```
        Point p2=new Point(20,30);
        System.out.println("p2.x="+p2.x);
        System.out.println("p2.y="+p2.y);
        Point p3=new Point(p2);
        System.out.println("p3.x="+p3.x);
        System.out.println("p3.y="+p3.y);
    }
}
```

运行结果为：

```
p1.x=0
p1.y=0
p2.x=20
p2.y=30
p3.x=20
p3.y=30
```

5．方法声明

方法声明包括方法名、返回类型和外部参数。其中参数的类型可以是简单数据类型，也可以是复合数据类型（又称引用数据类型）。

【例 1-6】　定义一个学生类，类名为 Student，它拥有两个实例变量分别为字符串类型变量 name 和整型变量 old，还拥有一个方法，方法名为 show，没有返回值和参数，功能是打印类中的成员变量的信息。

```
class Student {
    String name;
    int    age;
    void show(){
        System.out.println("my name is"+name);
        System.out.prntln("my old is"+age);
    }
}
```

1.6.6　继承性

通过继承实现代码复用。Java 中所有的类都是通过直接或间接地继承 java.lang.Object 类得到的。而 Object 类是所有类的祖先。继承而得到的类称为子类，被继承的类称为父类。子类不能继承父类中访问权限为 private 的成员变量和方法。子类可以重写父类的方法，及命名与父类同名的成员变量。但 Java 不支持多重继承，即一个类从多个超类派生的能力。

1．创建子类

格式：

```
class SubClassName extends SuperClassName {
…
}
```

说明：

SubClassName 标识符表示子类的名称。

SuperClassName 标识符表示父类的名称。

extends 关键字表示子类继承父类的关系。其实 extends 的含义是扩展。

【例 1-7】 继承实例演示。

```
//定义父类
class Point{
    int x, y;
    Point(int x, int y){
        this.x=x;
        this.y=y;
    }
    Point(){
        this.x=0;
        this.y=0;
    }
}
//定义子类
class Circle extends Point{
    int    radius;
    Circle(int r, int x, int y){
        radius=r;
        this.x=x;
        this.y=y;
    }
}
class CircleTest{
    public static void main(String args[]){
        Circle  c1=new Circle(200,50,50); //子类继承父类的成员变量 x 和 y
        System.out.println("圆的半径为:"+.c1.r);
        System.out.print("圆的圆心坐标为:");
        System.out.println("("+.c1.x+","+c1.y+")");
    }
}
```

运行结果为：

圆的半径为:200

圆的圆心坐标为:(50,50)

2．成员变量的隐藏和方法的覆盖

子类通过隐藏父类的成员变量和覆盖父类的方法，可以把父类的状态和行为改变为自身的状态和行为。

【例 1-8】 子类隐藏父类的成员变量实例说明。

```
class Father{
```

```
int x=5;
}
class Son extends Father{
int x=30; //隐藏了父类的变量 x
   public static void main(String args[]){
Son s1=new Son();
System.out.println("s1.x="+s1.x);
}
}
```

运行结果为：

```
s1.x=30
```

从此例看，子类对象 s1 能看到自身的成员变量 x，而看不到父类的同名变量 x，把这种现象称为子类隐藏父类的成员变量。

【例 1-9】 子类覆盖父类的方法实例说明。

```
class Father{
    void show(){
    System.out.println("Run show() method of Father");
   }
```

1.6.7 接口

Java 语言提供了一种接口（interface）机制。这种接口机制使 Java 的面向对象编程变得更加灵活。我们可以用接口来定义一个类的表现形式，但接口不能包含任何实现。在《Thinking in Java》一书中，作者对接口有这样的描述：“接口（interface）比抽象（abstract）的概念更进了一步。你可以把一个接口看成是一个纯的抽象类。”作者对接口的这一解释再准确不过了。

理解并用好接口机制将帮助我们更好的掌握 Java 这种面向对象的编程语言。下面来讨论一下接口的使用规则以及相关的应用。

1．接口的定义及实现

定义接口和定义类相似，只是要把 class 关键字换为 interface。定义方法时只需要方法名、返回类型和参数列表，不能有方法体。接口中可以定义字段，这些字段都被暗指为 static 和 final，因此应该根据需要先定好这些字段的值。例如：

```
public interface Flyable {
   void fly();
}
public interface Talkable {
   void talk();
}
public interface Message {
   int MAX_SIZE = 4096;
   String getMessage();
}
```

在上面定义的几个接口中，Flyable 和 Talkable 只定义了一个方法，而 Message 里除了方法外还有一个字段 MAX_SIZE。可以看出这些接口只定义了类的表现形式，而不包含任何实现，所以不能直接使用。要使用这些接口就需要有相应的类去实现它们。实现接口时应该先在类名后用 implements 关键字申明将要实现的接口，如果要实现多个接口，应该用逗号将它们隔开，然后一一实现这些接口中定义的方法。如下面的例子：

```
public class Parrot implements Flyable, Talkable {
   public void fly() {
      System.out.println("Flying like a parrot…");
   }
   public void talk() {
      System.out.println("Hello! I am a parrot!");
   }
}
public class MessageText implements Message {
   String message;
   public void setMessage(String msg) {
      message = msg;
      if (message.length() > MAX_SIZE)
         message = message.substring(0, MAX_SIZE);
      }
   public String getMessage() {
      return message;
     }
}
```

在 Parrot（鹦鹉）例子中，我们用接口 Flyable 来表示飞行能力，Talkable 表示说话能力，但它们并不包含具体实现。而 Parrot 同时具有这两种能力，所以为 Parrot 类同时实现了 Flyable 和 Talkable 两个接口。同样还可以定义一个 Swallow（燕子）类，但燕子只有飞行能力，所以只需要为 Swallow 实现 Flyable 就行了。因为它们各自的飞行方法有所不同，所以它们有各自关于飞行的具体实现。

另外，正因为一个类可以同时实现多个接口，使得 Java 的面向对象特性变得非常灵活。运用这种特性，可以实现类似 C++语言中多继承那样的特性，甚至更灵活的一些特性。下面来讨论一下接口在实际中的应用。

2．用接口来定义一些全局变量

因为接口内的字段都是 static 和 final 的，所以可以很方便的利用这一点来创建一些常量。例如：

```
public interface Constants {
   String ROOT = "/root";
   int MAX_COUNT = 200;
   int MIN_COUNT = 100;
}
```

在使用时可以直接用 Constants.ROOT 这样的形式来引用其中的常量。我们还可以用下面

这种方法来创建初始值不确定的常量。

```
public interface RandomColor {
int red = Math.random() * 255;
int green = Math.random() * 255;
int blue = Math.random() * 255;
}
```

其中，red、green 和 blue 的值会在第一次被访问时建立，然后保持不变。

3．用接口来定义基本数据结构

在设计一套软件系统的初期，可以用接口来对一些基本数据元素的特性进行一些描述，再根据需要进行不同的实现。请大家看看下面这个例子：

```
public interface User {
   int getAge();
   String getName();
   String getPassword();
}
public class XMLUser implements User {
   // 这里用 XML 技术实现 User 接口中的方法
   public int getAge() { ... }
   public String getName() { ... }
   public String getPassword() { ... }
}
public abstract class UserFactory {
   public static UserFactory getUserFactory() {
      return new XMLUserFactory();
   }
   public User getUser(String name);
   public User getAdmin();
   public User createUser(String name, String password, int age);
   public void addUser(User user);
   public void delUser(User user);
}
public class XMLUserFactory extends UserFactory {
   // 这里用 XML 技术实现 UserFactory 的抽象方法
}
```

在这个例子中，定义了一个接口 User 和一个抽象类 UserFactory。然后用 XML 技术实现这两个类。可以看出，只需要用 UserFactory 的 getUserFactory()就可以得到一个 UserFactory 的实例，而不用去考虑这个实例的具体实现方法。通过 UserFactory 的这个实例还可以直接得到 User 的实例，也不用去考虑具体的实现方法。

如果决定用 JDBC 技术来实现 User 和 UserFactory，只需要按上面的形式实现 JDBCUser 和 JDBCUserFactory 就行了。然后把 UserFactory 中的 getUserFactory 方法修改一下就可以改变它们的实现方法。而已经写好的调用 UserFactory 和 User 的部分不需要做任何修改。

这是用接口来定义数据结构的一个简单的例子，在实际应用中还有很多灵活的使用方法，

大家需要在学习过程中不断地去体会。

4．理解分布式应用的原理

目前有很多软件项目都使用了分布式的技术。Java 有多种支持分布式应用的技术，早期用的比较多的有 RMI、CORBA 等技术，而现在 EJB 技术更为流行一些。这些技术不管怎么发展，其实都是以接口为基础的。

以远程方法调用 RMI（Remote Method Invocation）为例。在编写 RMI 应用时，我们需要做两件最基本的事，首先要定义一个接口，这个接口要继承 java.rmi.Remote 接口，这个接口中应该包含要从远端调用的方法名。接下来是写一个类来实现这个接口中的方法。例如：

```
public interface Product extends java.rmi.Remote {
   String getName() throws java.rmi.RemoteException;
}
public class ProductImpl implements Product {
   String name;
   public ProductImpl(String n) {
      name = n;
   }
   public String getName() throws java.rmi.RemoteException {
      return name;
   }
}
```

在这个例子中，接口 Product 是放在客户端的，而 ProductImpl 是放在服务器端的，客户在使用时只需要用指定的规则得到 Product 的实例就行了，不用去考虑 Product 接口里的方法是如何实现的。在定义好这两个类后，用 Java 开发包命令“rmic ProductImpl”就可以帮助我们自动生成两个类 ProductImpl_Skel 和 ProductImpl_Stub。这两个类包含了 RMI 调用的运作机制。有兴趣的朋友可以把这两个类反编译后研究一下。你会发现其中 ProductImpl_Stub 实际上是接口 Product 的一个实现类。RMI 机制就是用这个类来生成 Product 的实例供客户端使用。另一个类 ProductImpl_Skel 则是在服务端响应 ProductImpl_Stub 的调用请求的类。而 RMI 最底层的通信原理则是利用 ObjectInputStream 和 ObjetOutputStream 通过 Socket 将要调用的方法名及参数列表传到服务器端，服务器端再通过特定的方法调用实现类（在本例中是 ProductImpl）的对应方法，然后将结果通过 Socket 传回客户端就行了。由于 Skel 和 Stub 类是用工具生成的，所以大大节省了开发的时间。另外，如果我们需要修改一些实现方法或错误，只需要对服务器端的实现类进行修改就可以了，也就是说这种分布式应用的大部分维护工作在服务器端就可以完成。

现在越来越多的应用使用了 EJB 技术。EJB 是从 RMI 发展而来的，它比 RMI 定义得更加完善，可以获得更好的面向对象的特性。但它的规则要比 RMI 复杂一些。不管它多复杂，它同样是使用了接口来定义各种不同的 Bean，也同样需要编写相应的实现类来完成具体的功能，最后还要通过 Socket 来进行通信。EJB 的运作机制本身有一定的复杂性，所以其应用的效率理所当然就会受到一定的影响。因此在选择开发技术时应该根据应用的规模和特点仔细考虑，不一定流行的技术就一定能适应你的应用。如果你很好的掌握了面向对象的设计原则，你就可以自行设计。也许可以根据自己应用的特点设计出更合适的分布式应用结构。

5．结论

除了上述的一些应用外，还有很多地方可以使用接口，比如在 Java 的事件机制中就常用到接口。另外，对于一些已经开发好的系统，在结构上进行较大的调整已经不太现实，这时可以通过定义一些接口并追加相应的实现来完成功能结构的扩展。

1.6.8 随机数生成函数

在 Java 中可以使用 java.util.Random 类来产生一个随机数发生器。它有两种形式的构造函数，分别是 Random()和 Random(long seed)。Random()使用当前时间即 System.currentTimeMillis()作为发生器的种子，Random(long seed)使用指定的 seed 作为发生器的种子。

随机数发生器即 Random 对象产生以后，可以通过对象调用不同的函数 nextInt()、nextLong()、nextFloat()、nextDouble()等来获得不同类型的随机数。

如果两个 Random 对象使用相同的种子（比如都是 100），并且以相同的顺序调用相同的函数，那么它们的返回值完全相同。如下面代码中两个 Random 对象的输出完全相同。

```
import java.util.*;
class RandomTest {
   public static void main(String[] args) {
      Random random1 = new Random(100);
      System.out.println(random1.nextInt());
      System.out.println(random1.nextInt());
      System.out.println(random1.nextFloat());
      System.out.println(random1.nextFloat());
      System.out.println(random1.nextBoolean());
      System.out.println(random1.nextBoolean());
      Random random2 = new Random(100);
      System.out.println(random2.nextInt());
      System.out.println(random2.nextInt());
      System.out.println(random2.nextFloat());
      System.out.println(random2.nextFloat());
      System.out.println(random2.nextBoolean());
      System.out.println(random2.nextBoolean());
   }
}
```

如果希望将返回的随机数控制在某个范围内（比如 0～99），则可以使用模数运算符%。说明：将模数运算符%作用于随机数产生器所产生的随机数身上，目的是为了让随机数的最大值局限于所制定的操作数数值减 1 范围内。如下面代码就将输入控制在 0～99 的范围内。

注意：如果不加 Math.abs()，输出范围将是−99～99。

```
import java.util.*;
class TestRandom {
   public static void main(String[] args) {
      Random random = new Random();
      for(int i = 0; i < 100; i++) {
```

```
            System.out.println(Math.abs(random.nextInt()) % 100);
        }
    }
}
```

1.7 实训

1.7.1 加法运算题

1．题目

随机产生两个 10 以内的数，显示这两个数相加运算，等待输入和，如果正确则输出“答对了”，如果不正确则输出“答错了”。

2．要求

（1）定义 Summand 类，取得 1～10 以内的加数。
（2）定义 Sum 类，显示加法运算，判断输入结果是否正确。
（3）定义 SumTest 类，进行测试。

1.7.2 员工涨工资

1．题目

设计一个接口用于涨工资，普通员工一次能涨 10%，经理能涨 20%。

2．要求

（1）定义 Man 类，有 name、address 两个属性，并写出构造方法。
（2）定义 Employee 类，有 employeeId（员工编号）、wage、workAge（工龄）三个属性，写出该类的构造方法。
（3）定义 Manager 类，有 leuel（级别）属性，写出构造方法。
（4）定义 EmployeeTest 类，产生一个员工和一个经理并输出其工资信息。

第 2 章　简单计算器

2.1　项目目标

利用 Java Swing 技术设计一个简单的计算器，项目运行界面如图 2-1 所示。能够进行两个数的加、减、乘、除运算。

图 2-1　项目运行界面

在图 2-1 所示的项目运行界面上实现如下功能：

（1）用户输入合法的数字后，选择相应的运算符，单击【计算】按钮，计算出相应的结果，并显示在对应的文本框中。

（2）如果在相应的文本框中没有输入数字，单击【计算】按钮，则弹出如图 2-2 所示的提示对话框。

图 2-2　提示对话框（一）

（3）如果在相应的文本框中输入的是非数字，单击【计算】按钮，则弹出如图 2-3 所示的提示对话框。

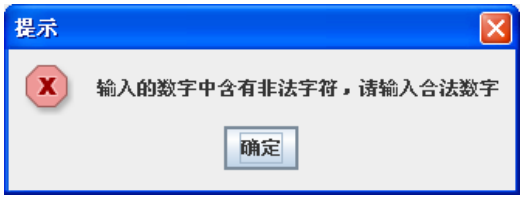

图 2-3　提示对话框（二）

（4）如果进行除法时，除数为 0，单击【计算】按钮，则弹出如图 2-4 所示的提示对话框。

图 2-4 提示对话框（三）

（5）操作者不能修改运算后的结果。

2.2　项目分析

项目的操作过程如下：

（1）在【第一操作数】文本框中输入计算的第一个数字。

（2）单击“+”执行加、“-”执行减、“*”执行乘或“/”执行除。

（3）在【第二操作数】文本框中输入计算的下一个数字。

（4）单击【计算】按钮，把结果送到对应的文本框中。

依据项目的操作过程描述，程序对应的算法如下：

（1）从界面获得操作数和运算符号。

（2）对操作数进行数据验证。

（3）把操作数转换成 double 数据类型。

（4）对操作数进行相应的计算。

（5）把结果显示在界面上。

依据程序设计思想，本项目将分为两层结构设计：视图层和业务模型层。视图层用来接受数据、显示结果、数据验证和调用业务逻辑等。业务模型层主要进行业务逻辑处理。如图 2-5 所示为项目模型设计。

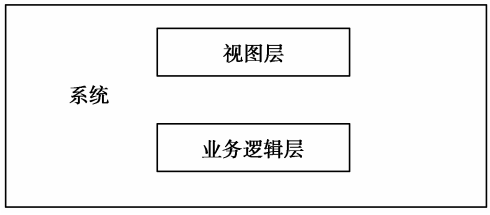

图 2-5　项目模型设计

根据分层设计思想，在视图层设计了两个类，在业务逻辑层设计了一个类。类的功能如表 2-1 所示。

表 2-1　类功能

序　号	类　名	功　能
1	CalFrame	负责视图层界面显示等功能
2	CheckData	负责视图层数据验证的功能
3	Calculate	负责业务逻辑层业务处理功能

Calculate 类是对业务逻辑的抽象，其中包含一个 cal 方法，功能是对两个数据进行加、减、乘、除计算，格式如下：

```
public double cal(double d1,double d2,String op)
```

其中，参数 d1 和 d2 代表运算数，参数 op 代表运算符。判断参数 op 的内容，决定进行相应的计算。

CheckData 类是对数据进行验证，其中包含一个 check 方法，功能是判断数据是否为空，是否含有非数字字符等。格式如下：

```
public boolean check(String s)
```

对于判断是否为数字，有两种方法：

第一种方法是对字符串的每一位字符进行判断是否是 0～9 或是“•”字符，不是则跳出判断，标志此字符串中含有非法字符。

第二种方法是利用 Doulble 类提供的 parseDouble 方法来判断是否是数字。即把字符串转换成 doulbe 类型数据，含有非法字符，该方法将抛出异常。

CalFrame 类是用来搭建界面，并调用 CheckData、Calculate 对象。该类继承了 JFame，实现了 ActionListener 接口，拥有 CalFrame()、jbInit()、main(String[] args)、actionPerformed(ActionEvent e)方法。CalFrame()方法是构造方法，负责调用 jbInit()；jbInit()方法主要是在 Frame 上加载各种组件，设置 Frame 大小等功能；actionPerformed(ActionEvent e)方法实现命令按钮单击事件应完成的动作；main(String[] args)方法用于运行 CalFrame 类，即本项目。CalFrame 拥有的属性如表 2-2 所示。

表 2-2　CalFrame 类属性表

序　号	属性类型（组件名）	属 性 名 称	备　注
1	Jlabel	lbl_l1	显示操作数一
2	Jlabel	lbl_l2	显示操作数二
3	Jlabel	lbl_l3	显示计算结果
4	JtextField	txt_d1	接收操作数
5	JtextField	txt_d2	接收操作数
6	JtextField	txt_result	接收结果
7	JradioButton	rdb_r1	单选按钮显示+
8	JradioButton	rdb_r2	单选按钮显示-

续表

序　　号	属性类型（组件名）	属 性 名 称	备　　注
9	JradioButton	rdb_r3	单选按钮显示*
10	JradioButton	rdb_r4	单选按钮显示/
11	ButtonGroup	ButtonGroup1	单选按钮组
12	Jbutton	btn_cal	计算按钮

CalFrame 类界面设计如图 2-6 所示。

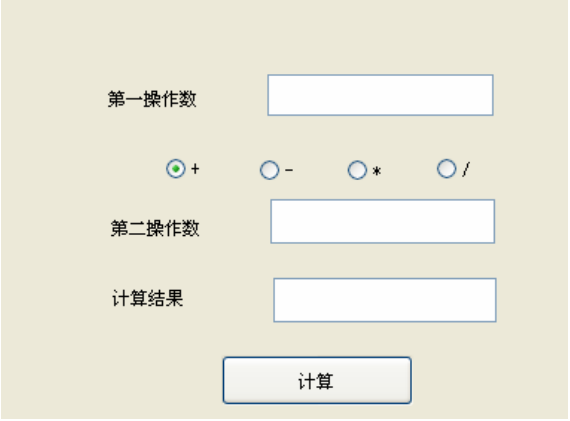

图 2-6　项目设计界面

2.3　代码实现

Calculate.java 文件源码：

```
package cvit.com.cn;
public class Calculate {
    //进行业务处理，该方法实现加、减、乘、除功能
    public double cal(double d1,double d2,String op){
        double rel=0;
        if(op.equals("+"))
            rel=d1+d2;
        else if(op.equals("-"))
            rel=d1-d2;
        else if(op.equals("*"))
            rel=d1*d2;
        else
            if(d2==0)
                rel=0;
            else
                rel=d1/d2;
            return rel;
    }
```

```
    }
    //CheckData.java
    package cvit.com.cn;
    import javax.swing.JOptionPane;
    public class CheckData {
        //对字符串进行验证
        public boolean check(String s){
            if (s.equals("")) {
                JOptionPane.showMessageDialog(null,"不能为空，请输入合法数字","提示",JOption
Pane.ERROR_MESSAGE);
                return false;
            } else {
                try {
                    double d = Double.parseDouble(s);
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(null, "输入的数字中含有非法字符，请输入合法
数字","数据错误", JOptionPane.ERROR_MESSAGE);
                    return false;
                }
            }
            return true;
        }
    }
```

CalFrame.java 文件源码：

```
    package cvit.com.cn;
    import java.awt.BorderLayout;
    import javax.swing.JFrame;
    import javax.swing.JLabel;
    import java.awt.Rectangle;
    import javax.swing.JTextField;
    import javax.swing.JButton;
    import javax.swing.JRadioButton;
    import javax.swing.ButtonGroup;
    import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
    import javax.swing.JOptionPane;
    public class CalFrame extends JFrame implements ActionListener {
        public CalFrame() {
            try {
                jbInit();
            } catch (Exception exception) {
                exception.printStackTrace();
            }
        }
```

```
private void jbInit() throws Exception {
        getContentPane().setLayout(null);
        lbl_l1.setText("第一操作数");
        lbl_l1.setBounds(new Rectangle(78, 60, 105, 31));
        txt_d2.setBounds(new Rectangle(187, 143, 151, 30));
        txt_d1.setBounds(new Rectangle(185, 59, 152, 28));
        btn_cal.setBounds(new Rectangle(154, 248, 129, 34));
        btn_cal.setText("计算");

        btn_cal.addActionListener(this);
        txt_result.setEnabled(false);
        txt_result.setBounds(new Rectangle(189, 196, 150, 30));
        lbl_l3.setText("计算结果");
        lbl_l3.setBounds(new Rectangle(81, 196, 74, 26));
        rdb_r1.setText("+");
        rdb_r1.setBounds(new Rectangle(113, 111, 44, 23));
        rdb_r2.setText("−");
        rdb_r2.setBounds(new Rectangle(176, 112, 48, 23));
        rdb_r3.setText("*");
        rdb_r3.setBounds(new Rectangle(235, 112, 43, 23));
        rdb_r4.setText("/");
        rdb_r4.setBounds(new Rectangle(295, 110, 38, 24));
        this.getContentPane().add(lbl_l1);
        this.getContentPane().add(lbl_l3);
        this.getContentPane().add(txt_result);
        this.getContentPane().add(lbl_l2);
        this.getContentPane().add(txt_d2);
        this.getContentPane().add(txt_d1);
        this.getContentPane().add(rdb_r3);
        this.getContentPane().add(rdb_r2);
        this.getContentPane().add(rdb_r1);
        this.getContentPane().add(rdb_r4);
        this.getContentPane().add(btn_cal);
        lbl_l2.setText("第二操作数");
        lbl_l2.setBounds(new Rectangle(80, 146, 87, 32));
        rdb_r1.setSelected(true);
        buttonGroup1.add(rdb_r1);
        buttonGroup1.add(rdb_r2);
        buttonGroup1.add(rdb_r3);
        buttonGroup1.add(rdb_r4);
        this.setTitle("简单计算器");
        this.setSize(450,350);
        this.setVisible(true);
}
public static void main(String[] args) {
        CalFrame calframe = new CalFrame();
```

```
}
JLabel lbl_l1 = new JLabel();
JLabel lbl_l2 = new JLabel();
JTextField txt_d1 = new JTextField();
JTextField txt_d2 = new JTextField();
JLabel lbl_l3 = new JLabel();
JTextField txt_result = new JTextField();
JButton btn_cal = new JButton();
JRadioButton rdb_r1 = new JRadioButton();
JRadioButton rdb_r2 = new JRadioButton();
JRadioButton rdb_r3 = new JRadioButton();
JRadioButton rdb_r4 = new JRadioButton();
ButtonGroup buttonGroup1 = new ButtonGroup();
ButtonGroup buttonGroup2 = new ButtonGroup();
public void actionPerformed(ActionEvent e) {
    //获得数据
    String d1 = txt_d1.getText();
    String d2 = txt_d2.getText();
    String op = "";
    if (rdb_r1.isSelected()) {
        op = rdb_r1.getActionCommand();
    } else if (rdb_r2.isSelected()) {
        op = rdb_r2.getActionCommand();
    } else if (rdb_r3.isSelected()) {
        op = rdb_r3.getActionCommand();
    } else {
        op = rdb_r4.getActionCommand();
    }
    //验证数据
    CheckData cd = new CheckData();
    boolean flag;
    flag = cd.check(d1);
    if (!flag) {
        txt_d1.setText("");
        txt_d1.requestFocus();
        return;
    }
    flag = cd.check(d2);
  if (!flag) {
      txt_d2.setText("");
      txt_d2.requestFocus();
      return;
  }
  //验证除数是 0 问题
  if(d2.equals("0") && op.equals("/")){
      JOptionPane.showMessageDialog(null, "除数不能为 0，请输入合法数字",
        "提示",JOptionPane.ERROR_MESSAGE);
```

```
                return;
            }
            //将数据转换成 double 类型
            double data1=Double.parseDouble(d1);
            double data2=Double.parseDouble(d2);
            //进行计算
            Calculate calculate=new Calculate();
            double result=calculate.cal(data1,data2,op);
            //把结果转换成 String 类型
            txt_result.setText(String.valueOf(result));
        }
    }
```

2.4 运行与发布

2.4.1 运行

将 Calculate.java、CheckData.java、CalFrame.java 3 个文件保存到一个文件夹中，如 e:\java\cal。在使用 javac 命令进行编译之前，应使用如下命令设置类路径：

```
e:\java\cal＞set classpath= e:\java\cal
```

然后利用 javac 命令对文件进行编译，使用如下命令：

```
Javac CalFrame.java
```

之后，使用 java 执行程序：

```
Java CalFrame
```

程序即运行。

2.4.2 发布

使用 jar.exe 将应用程序打包，把应用程序中涉及的类和图片压缩成一个 jar 文件，这样就可以发布程序了。

（1）编写一个程序文件，名称为 MANIFEST.MF，代码如下：

```
Manifest –Version: 1.0
Created –By: 1.5.0_02 (Sun Microsystems Inc.)
Main-Class: CalFrame
```

MANIFEST.MF 文件保存在 e:\java\ calculator。

（2）使用如下命令生成 jar 文件：

```
jar cfm CalFrame.jar MANIFEST.MF *.class
```

（3）编写一个 bat 文件，文件名为 CalFrame.bat。

内容为：javaw –jar CalFrame.jar

与 CalFrame.jar 保存在同一个文件夹下。

（4）运行 CalFrame.jar 文件即可。前提是计算机上安装了 Java JDK，并配置了环境变量。

2.5 本项目实现中常见问题

（1）实现 ActionListener 接口时，一定要重写 public void actionPerformed(ActionEvent e) 方法。

（2）命令按钮要进行事件注册。

（3）要设置 frame 的大小和可见性，否则运行时看不到窗体。

（4）向 frame 中添加组件时，要添加到 frame 的内容窗格中。

（5）把 Sring 类型数据转换成 double 类型用 Double.parseDouble(String s)方法，不能用强制类型转换 double d1 =(double)s;。

（6）建议字符串判断内容相等用 equals 方法。

2.6 项目技术支持

2.6.1 Swing简介

Swing 是 Java 语言编写 GUI（Graphics User Interface，用户图形界面）的新技术，它是在 AWT（Abstract Window Toolkit，抽象窗口工具包）基础上发展起来的。Swing 提供了许多开发包，极大地丰富了 Java 的图形界面功能。AWT 设计的初衷是支持开发小应用程序的简单用户界面，对图形界面支持不全面，缺少剪贴板、打印支持、键盘导航、弹出式菜单、滚动窗格等元素。

随着发展的需要，出现了 Swing，Swing 组件几乎都是轻量组件，与重量组件相比，没有本地的对等组件，不像重量组件要在它们自己的本地不透明窗体中绘制，轻量组件在它们的重量组件的窗口中绘制。

Swing 由纯 Java 技术实现，Swing 组件是用 Java 实现的轻量级（light-weight）组件，没有本地代码，不依赖操作系统的支持，这是它与 AWT 组件的最大区别。由于 AWT 组件通过与具体平台相关的对等类（Peer）实现，因此 Swing 比 AWT 组件具有更强的实用性。Swing 在不同的平台上表现一致，并且有能力提供本地窗口系统不支持的其他特性。

Swing 采用了一种 MVC 的设计模式，即“模型-视图-控制”（Model-View-Controller），其中模型用来保存内容，视图用来显示内容，控制器用来控制用户输入。

在 javax.swing 包中，定义了两种类型的组件：顶层容器（JFrame，JApplet，JDialog 和 JWindow）和轻量级组件。Swing 组件都是 AWT 的 Container 类的直接子类和间接子类。

```
java.awt.Component
    -java.awt.Container
        -java.awt.Window
            -java.awt.Frame-javax.swing.JFrame
            -javax.Dialog-javax.swing.JDialog
            -javax.swing.JWindow
        -java.awt.Applet-javax.swing.JApplet
        -javax.swing.Box
        -javax.swing.Jcomponet
```

Swing 包是 JFC（Java Foundation Classes）的一部分，由许多包组成，如表 2-3 所示。

表 2-3　Swing 包中的类描述

序　号	包	描　述
1	com.sum.swing.plaf.motif	用户界面代表类，它们实现 Motif 界面样式
2	com.sum.java.swing.plaf.windows	用户界面代表类，它们实现 Windows 界面样式
3	javax.swing	Swing 组件和使用工具
4	javax.swing.border	Swing 轻量组件的边框
5	javax.swing.colorchooser	JcolorChooser 的支持类/接口
6	javax.swing.event	事件和监听器类
7	javax.swing.filechooser	JFileChooser 的支持类/接口
8	javax.swing.pending	未完全实现的 Swing 组件
9	javax.swing.plaf	抽象类，定义 UI 代表的行为
10	javax.swing.plaf.basic	实现所有标准界面样式公共功能的基类
11	javax.swing.plaf.metal	用户界面代表类，它们实现 Metal 界面样式
12	javax.swing.table	Jtable 组件
13	javax.swing.text	支持文档的显示和编辑
14	javax.swing.text.html	支持显示和编辑 HTML 文档
15	javax.swing.text.html.parser	Html 文档的分析器
16	javax.swing.text.rtf	支持显示和编辑 RTF 文件
17	javax.swing.tree	Jtree 组件的支持类
18	javax.swing.undo	支持取消操作

javax.swing 包是 Swing 提供的最大包，它包含将近 100 个类和 25 个接口，几乎所有的 Swing 组件都在 javax.swing 包中，只有 JTableHeader 和 JTextComponent 例外，它们分别在 swing.table 和 swing.text 中。

javax.swing.event 包中定义了事件和事件监听器类，与 AWT 的 event 包类似。它们都包括事件类和监听器接口。

javax.swing.pending 包包含了没有完全实现的 Swing 组件。

javax.swing.table 包中主要包括表格组建（JTable）的支持类。

javax.swing.tree 同样是 JTree 的支持类。

javax.swing.text、javax.swing.text.html、javax.swing.text.html.parser 和 javax.swing.text.rtf 都是用于显示和编辑文档的包。

2.6.2　项目涉及的Swing组件

1．JFrame（框架）组件

JFrame 类是由 Frame 类扩展而来的，主要是为其他组件的绘制提供位置。

（1）构造方法：

JFrame() 创建一个不可见的新窗体。

JFrame(GraphicsConfiguration gc) 以屏幕设备的指定 GraphicsConfiguration 和空白标题创建一个新窗体。

JFrame(String title) 创建一个新的、初始不可见的、具有指定标题的新窗体。

JFrame(String title, GraphicsConfiguration gc) 创建一个具有指定标题和指定屏幕设备的 GraphicsConfiguration 的新窗体。

（2）常用方法：

Component add(Component comp) 将指定组件追加到此容器的尾部。

void setTitle(String title) 设置窗体的标题。

String getTitle() 获得窗体的标题。

void setSize(int width, int height) 设置窗体的大小。

int getHeight() 获得窗体的当前高度。

int getWidth() 获得窗体的当前宽度。

int getX() 获得窗体的当前 x 坐标。

int getY() 获得窗体的当前 y 坐标。

Dimension getSize() 以 Dimension 对象的形式获得窗体的大小。Dimension 对象的 height 字段包含此组件的高度，而 Dimension 对象的 width 字段则包含此组件的宽度。

void setJMenuBar(JMenuBar menubar) 设置此窗体的菜单栏。

void setLayout(LayoutManager manager) 设置窗体的布局管理器。

void setVisible(boolean flag) 设置窗体的可见性。

2．JLable（标签）组件

JLable 是不可编辑的显示区域，可以容纳文字、图像等内容形式。

（1）构造方法：

JLabel() 创建无图像并且其标题为空字符串的标签。

JLabel(Icon image) 创建具有指定图像的标签。

JLabel(Icon image, int horizontalAlignment) 创建具有指定图像和水平对齐方式的标签。

JLabel(String text) 创建具有指定文本的标签。

JLabel(String text, Icon icon, int horizontalAlignment) 创建具有指定文本、图像和水平对齐方式的标签。

JLabel(String text, int horizontalAlignment) 创建具有指定文本和水平对齐方式的标签。

（2）常用方法：

String getText() 获得标签所显示的内容的字符串。

void setText(String text) 设置标签所显示的内容。

Icon getIcon() 获得标签显示的图形图像（字形、图标）。

void setIcon(Icon icon) 设置标签显示的图形图像。

3．JTextField（文本框）组件

JTextField 是一个轻量级组件，允许编辑单行文本。

（1）构造方法：

JTextField() 创建一个新的空文本框。

JTextField(Document doc, String text, int columns) 创建一个新的文本框，并给定文本存储模型和列数。

JTextField(int columns) 创建一个具有指定列数的新的空文本框。

JTextField(String text) 创建一个含有指定内容的文本框。

JTextField(String text, int columns) 创建一个含有指定内容和列数的文本框。

（2）常用方法：

void addActionListener(ActionListener actionlistener) 添加指定的操作侦听器以从此文本字段接收操作事件。

void setText(String t) 设置文本框显示的内容。

String getText() 获得文本框显示的内容。

void setSelectEnd(int) 将选择结束点设置为指定的位置。

void setSelectStart(int) 将选定起始点设置为指定的位置。

String getSelectText() 获得选中文本框的内容。

void requestFocus() 设置文本框具有输入焦点。

4．JButton（命令按钮）组件

（1）构造方法：

JButton() 创建不带有设置文本或图标的按钮。

JButton(Action a) 创建一个按钮，其属性从所提供的 Action 中获取。

JButton(Icon icon) 创建一个带图标的按钮。

JButton(String text) 创建一个带文本的按钮。

JButton(String text, Icon icon) 创建一个带初始文本和图标的按钮。

（2）常用方法：

void addActionListener(ActionListener al) 设置 al 对按钮进行监听。

String getActionCommand() 获得按钮的动作命令。

String getText() 获得按钮的文本。

void setActionCommand(String command) 设置按钮的动作命令。

void setText(String text) 设置按钮的文本。

5．JRadioButton（单选按钮）组件

JRadioButton 提供可被选择或取消选择的按钮，并可为用户显示其状态。

（1）构造方法：

JRadioButton() 创建一个初始化为未选择的单选按钮，其文本未设定。

JRadioButton(Action a) 创建一个单选按钮，其属性来自提供的 Action。

JRadioButton(Icon icon) 创建一个初始化为未选择的单选按钮，其具有指定的图像但无文本。

JRadioButton(Icon icon, boolean selected) 创建一个具有指定图像和选择状态的单选按钮，但无文本。

JRadioButton(String text) 创建一个具有指定文本的状态为未选择的单选按钮。

JRadioButton(String text, boolean selected) 创建一个具有指定文本和选择状态的单选按钮。

JRadioButton(String text, Icon icon) 创建一个具有指定的文本和图像并初始化为未选择的单选按钮。

JRadioButton(String text, Icon icon, boolean selected) 创建一个具有指定的文本、图像和选择状态的单选按钮。

（2）常用方法：

String getText() 获得单选按钮的文本内容。

void setText(String text) 设置单选按钮的文本内容。

boolean isSelected() 返回按钮的状态。如果选定了单选按钮，则返回 true，否则返回 false。

void setSelected(boolean b) 设置按钮的状态。

6．JButtonGroup（按钮组）

JButton Group 用于为一组按钮创建一个多斥作用域。使用相同的按钮对象创建一组，按钮组意味着“开启”其中一个按钮时，将关闭组中的其他所有按钮。

（1）构造方法：

ButtonGroup()创建一个新的按钮组。

（2）常用方法：

void add(AbstractButton b) 将按钮添加到组中。

int getButtonCount() 获得组中的按钮数。

Enumeration<AbstractButton> getElements() 获得组中的所有按钮。

ButtonModel getSelection() 获得选择按钮的模型。

boolean isSelected(ButtonModel m) 返回对是否已选择一个 ButtonModel 的判断。

void remove(AbstractButton b) 从组中移除按钮。

void setSelected(ButtonModel m, boolean b) 为 ButtonModel 设置选择值。

7．JOptionPane（对话框）

JOptionPane 类用于弹出要求用户提供值或向其发出通知的标准对话框，包括确认对话框、输入对话框、消息对话框等。

（1）构造方法：

JOptionPane() 创建一个带有测试消息的对话框。

JOptionPane(Object message) 创建一个显示消息的对话框，使其使用 UI 提供的普通消息类型和默认选项。

JOptionPane(Object message, int messageType) 创建一个显示消息的对话框，使其具有指定的消息类型和默认选项。

JOptionPane(Object message, int messageType, int optionType) 创建一个显示消息的对话框，使其具有指定的消息类型和选项。

JOptionPane(Object message, int messageType, int optionType, Icon icon) 创建一个显示消息的 JOptionPane 的实例，使其具有指定的消息类型、选项和图标。

JOptionPane(Object message, int messageType, int optionType, Icon icon, Object[] options)

创建一个显示消息的对话框，使其具有指定的消息类型、图标和选项。

JOptionPane(Object message, int messageType, int optionType, Icon icon, Object[] options, Object initialValue) 在指定最初选择的选项的前提下，创建一个显示消息的对话框，使其具有指定的消息类型、图标和选项。

（2）常用方法：

static int showConfirmDialog(Component parentComponent, Object message) 调出带有选项 Yes、No 和 Cancel 的对话框；标题为 Select an Option。

static int showConfirmDialog(Component parentComponent, Object message, String title, int optionType)调出一个由 optionType 参数确定其中选项数的对话框。

static int showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType) 调用一个由 optionType 参数确定其中选项数的对话框，messageType 参数确定要显示的图标。

static int showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon) 调出一个带有指定图标的对话框，其中的选项数由 optionType 参数确定。

static String showInputDialog(Component parentComponent, Object message) 显示请求用户输入内容的问题消息对话框，它以 parentComponent 为父级。

static String showInputDialog(Component parentComponent, Object message, Object initialSelectionValue) 显示请求用户输入内容的问题消息对话框，它以 parentComponent 为父级。

static String showInputDialog(Component parentComponent, Object message, String title, int messageType) 显示请求用户提供输入的对话框，它以 parentComponent 为父级，该对话框的标题为 title，消息类型为 messageType。

static Object showInputDialog(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object initialSelectionValue) 提示用户在可以指定初始选择、可能选择及其他所有选项的模块化的对话框中输入内容。

static String showInputDialog(Object message) 显示请求用户输入的问题消息对话框。

static String showInputDialog(Object message, Object initialSelectionValue) 显示请求用户输入的问题消息对话框，它带有已初始化为 initialSelectionValue 的输入值。

static void showMessageDialog(Component parentComponent, Object message) 调出标题为“Message”的信息消息对话框。

static void showMessageDialog(Component parentComponent, Object message, String title, int messageType) 调出对话框，它显示使用由 messageType 参数确定的默认图标的 message。

static void showMessageDialog(Component parentComponent, Object message, String title, int messageType, Icon icon) 调出一个显示信息的对话框，为其指定了所有参数。

static int showOptionDialog(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon, Object[] options, Object initialValue) 调出一个带有指定图标的对话框，其中的初始选择由 initialValue 参数确定，选项数由 optionType 参数确定。

（3）常用方法中参数解释：

ParentComponent：设置对话框的父窗口对象，一般为当前窗口。也可以为 null 即采用默

认的 Frame 作为父窗口，此时对话框将设置在屏幕的正中。

message：设置要在对话框内显示的描述性的文字。

String title：设置对话框标题。

Component：设置对话框内要显示的组件（如按钮）。

Icon：设置对话框内要显示的图标。

messageTyp：设置对话框显示的图标，一般可以设置的值如表 2-4 所示。

表 2-4　messageTyp 可选值

序　号	值	图 标 样 式
1	ERROR_MESSAGE	
2	INFORMATION_MESSAGE	
3	WARNING_MESSAGE	
4	QUESTION_MESSAGE	
5	PLAIN_MESSAGE	

optionType：设置在对话框上显示的按钮选项，一般可以设置的值如表 2-5 所示。

表 2-5　optionTyp 可选值

序　号	值	含　义
1	DEFAULT_OPTION	在对话框上显示【确定】按钮
2	YES_NO_OPTION	在对话框上显示【是】和【否】按钮
3	YES_NO_CANCEL_OPTION	在对话框上显示【是】、【否】和【撤销】按钮
4	OK_CANCEL_OPTION	在对话框上显示【确定】和【撤销】按钮

（4）showXxxDialog 方法返回值。在对话框单击相应的按钮返回的值如表 2-6 所示。

表 2-6　optionTyp 可选值

序　号	值	含　义
1	YES_OPTION	在对话框上单击【是】按钮，返回的值
2	NO_OPTION	在对话框上单击【否】按钮，返回的值
3	CANCEL_OPTION	在对话框上单击【撤销】按钮，返回的值
4	OK_OPTION	在对话框上单击【确定】按钮，返回的值
5	CLOSED_OPTION	用户没有做出任何选择而关闭了对话框返回的值

（5）对话框举例。

① 消息对话框：

```
JOptionPane.showMessageDialog(null," 除 数 不 能 为  0 ， 请 输 入 合 法 数 字 "," 提 示
```

",JOptionPane.ERROR_MESSAGE);

显示对话框如图 2-7 所示。

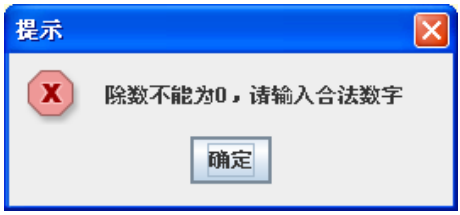

图 2-7 “消息”对话框

② 输入对话框：

```
String  firstNumber = JOptionPane.showInputDialog( "请输入第 1 个整数" );
```

输入对话框如图 2-8 所示。

图 2-8 “输入”对话框

③ 确认对话框：

```
int a=JOptionPane.showConfirmDialog(null,"是否删除","提示",JOptionPane.YES_NO_OPTION);
```

确认对话框如图 2-9 所示。

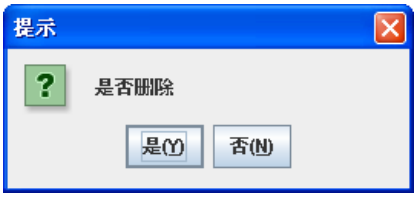

图 2-9 “确认”对话框

2.6.3 事件处理机制

Java 处理事件中涉及的事件类和接口都在 java.awt.event 包下。

1．Java事件处理机制中的概念

事件：一个事件类型的对象，用来描述发生了什么事，当用户在组件上进行操作时会触发相应的事件。如进行鼠标操作，则产生 MouseEvent 事件。

事件源：能够产生事件的 GUI 组件，如 JButton 按钮、JTextField 文本框组件等。

事件处理者：接收事件对象并对其进行处理的对象，也被成为事件监听器。

事件从事件源到监听器的传递是通过对目标监听者对象的 Java 方法调用进行的。对每个明确的事件的发生，都相应地定义一个明确的 Java 方法。这些方法都集中定义在事件监听器（EventListener）接口中，这个接口要继承 java.util.EventListener。实现了事件监听器接口中一些或全部方法的类就是事件监听器。伴随着事件的发生，相应的状态通常都封装在事件状态对象中，该对象必须继承 java.util.EventObject。事件状态对象作为单参传递给应响应该事件的监听者方法中。发出某种特定事件的事件源的标识是：遵从规定的设计格式为事件监听者定义注册方法，并接受对指定事件监听者接口实例的引用。

2．使用授权处理模型进行事件处理的一般方法

对于某种类型的事件 XxxEvent，要想接收并处理这类事件，必须定义相应的事件处理类，该类要实现与事件相对应的接口 XxxListener，即定义事件处理类并实现监听器接口。

事件源事例化以后，必须进行授权，注册该类事件的监听器。使用 addXxxListener（XxxListener）方法来注册监听器。

在事件处理类中重写其事件处理的方法体。事件源、事件类和监听器之间的关系如图 2-10 所示。

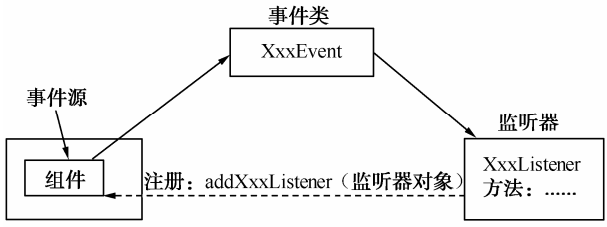

图 2-10　事件处理模型

3．Java中常见的事件类和对应的事件接口（如表 2-7 所示）

表 2-7　常见事件和对应的事件接口

事　件　名	对应的接口	接口中包含的抽象的方法
ActionEvent	ActionListener	public void actionPerformed(ActionEvent e)

FocusEvent	FocusListener	//获得焦点
		void focusGained(FocusEvent e)
		//失去焦点
		void focusLost(FocusEvent e)

KeyEvent	KeyListener	//当键被按下
		pulic void keyPressed(KeyEvent e);
		//当键被松开
		public void keyReleased(KeyEvent e);
		//当键被击打
		public void keyTyped(KeyEvent e);

续表

事　件　名	对应的接口	接口中包含的抽象的方法

MouseEvent	MouseListener	//鼠标单击
		public void mouseClicked(MouseEvent e);
		//鼠标进入
		public void mouseEntered(MouseEvent e);
		//鼠标离开
		public void mouseExited(MouseEvent e);
		//鼠标按下
		public void mousePressed(MouseEvent e);
		//鼠标释放
		public void mouseReleased(MouseEvent e);

	MouseMotionListener	//按下键后拖动鼠标
		public void mouseDragged(MouseEvent e);
		//移动鼠标
		public void mouseMoved(MouseEvent e);

TextEvent	TextListener	//当文本框中的内容发生变化产生 TextEvent 事件 public void textValueChanged(TextEvent e)
WindowEvent	WindowListener	void windowActivated(WindowEvent e) void windowClosed(WindowEvent e) void windowClosing(WindowEvent e) void windowDeactivated(WindowEvent e) void windowDeiconified(WindowEvent e) void windowIconified(WindowEvent e) void windowOpened(WindowEvent e)

4．实例

单击窗体上的红色按钮，使窗体的颜色为红色；单击窗体上的蓝色按钮，使窗体的颜色为蓝色，如图 2-11 所示。

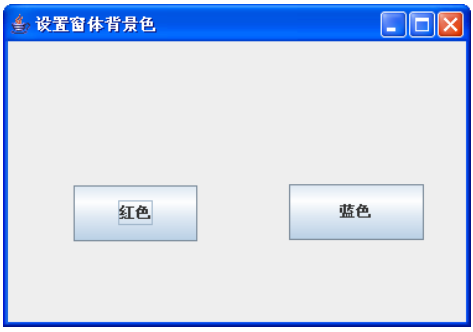

图 2-11　实例界面

本实例通过自身类作为事件监听器、外部类作为事件监听器、匿名内部类作为事件监听器和内部类作为事件监听器 4 种方式来实现。读者根据实际情况选择相应事件处理方式。

实现代码如下：

```
//自身类作为事件监听器
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.Color;
public class SetFrame extends JFrame implements ActionListener{
    JButton btn_red = new JButton();
    JButton btn_blue = new JButton();
    public SetFrame () {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        getContentPane().setLayout(null);
        btn_red.setBounds(new Rectangle(53, 116, 100, 45));
        btn_red.setText("红色");
        //按命令进行注册
        btn_red.addActionListener(new Frame1());
        btn_blue.addActionListener(new Frame1());
        this.getContentPane().add(btn_red);
        this.getContentPane().add(btn_blue);
        btn_blue.setBounds(new Rectangle(227, 115, 109, 45));
        btn_blue.setText("蓝色");
        this.setTitle("设置窗体背景色");
        this.setSize(300,300);
        this.setVisible(true);
    }

    public static void main(String[] args) {
        SetFrame frame1 = new SetFrame ();
    }
    //实现 ActionListener 接口方法
    public void actionPerformed(ActionEvent e) {
        if(e.getSource()==btn_red){
            this.getContentPane().setBackground(Color.red);
        }
        if(e.getSource()==btn_blue){
            this.getContentPane().setBackground(Color.blue);
```

```
        }

    }
}
```

运行效果如图 2-12 所示。

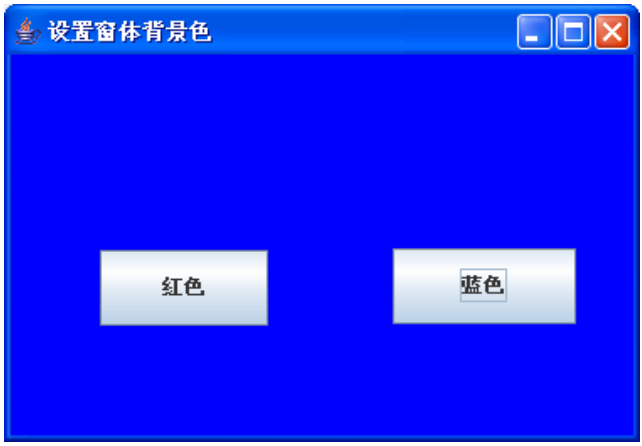

图 2-12　实例运行效果

btn_red 和 btn_blue 对象（按钮）是事件源；btn_red 和 btn_blue 事件源产生 ActionEvent 事件，被 Frame1 监听器捕获到，该对象调用 actionPerformed（ActionEvent e）方法进行事件处理，其中 e 对象就是对产生的事件相关信息进行的封装。Frame1 类成为事件处理者是因为按钮注册的 ActionListener 事件处理器是 Frame1 生成的对象，而且 Frame1 类也实现了 ActionListener 接口。

```
//外部类作为事件监听器
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.*;
import java.awt.datatransfer.Clipboard;
public class Frame1 extends JFrame {
    JButton btn_red = new JButton();
    JButton btn_blue = new JButton();
    public Frame1() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }

    private void jbInit() throws Exception {
```

```
            getContentPane().setLayout(null);
            btn_red.setBounds(new Rectangle(53, 116, 100, 45));
            btn_red.setText("红色");
            //创建事件处理类对象
            OuterClass oc=new OuterClass(this);
            //对命令按钮进行注册
            btn_red.addActionListener(oc);
            btn_blue.addActionListener(oc);
            this.getContentPane().add(btn_red);
            this.getContentPane().add(btn_blue);
            btn_blue.setBounds(new Rectangle(227, 115, 109, 45));
            btn_blue.setText("蓝色");
            this.setTitle("设置窗体背景色");
            this.setSize(300,300);
            this.setVisible(true);
        }

        public static void main(String[] args) {
            Frame1 frame1 = new Frame1();
        }

    }

    /*外部类实现 ActionListener 接口*/
    class OuterClass implements ActionListener{
        Frame1 oce;

        public OuterClass(Frame1 oce){
            this.oce = oce;
        }

        public void actionPerformed(ActionEvent e){
            Container c=oce.getContentPane();
            if(e.getSource()==oce.btn_blue)
                c.setBackground(Color.blue);
           if(e.getSource()==oce.btn_red)
               c.setBackground(Color.red);
        }
    }

    //匿名内部类作为事件监听器
    import javax.swing.JFrame;
    import javax.swing.JButton;
    import java.awt.Rectangle;
    import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
    import java.awt.*;
```

```
import java.awt.datatransfer.Clipboard;
public class Frame1 extends JFrame {
    JButton btn_red = new JButton();
    JButton btn_blue = new JButton();
    public Frame1() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }

    private void jbInit() throws Exception {
        getContentPane().setLayout(null);
        btn_red.setBounds(new Rectangle(53, 116, 100, 45));
        btn_red.setText("红色");
       //匿名类实现事件监听
        btn_red.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                    Container c=getContentPane();
                    c.setBackground(Color.red);
                }
            }
        );

        btn_blue.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                    Container c=getContentPane();
                    c.setBackground(Color.blue);
                }
            }
        );

        this.getContentPane().add(btn_red);
        this.getContentPane().add(btn_blue);
        btn_blue.setBounds(new Rectangle(227, 115, 109, 45));
        btn_blue.setText("蓝色");
        this.setTitle("设置窗体背景色");
        this.setSize(300,300);
        this.setVisible(true);
    }

    public static void main(String[] args) {
        Frame1 frame1 = new Frame1();
    }
}
```

```
//内部类作为事件监听器
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.*;
import java.awt.datatransfer.Clipboard;

public class Frame1 extends JFrame {
    JButton btn_red = new JButton();
    JButton btn_blue = new JButton();
    public Frame1() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        getContentPane().setLayout(null);
        btn_red.setBounds(new Rectangle(53, 116, 100, 45));
        btn_red.setText("红色");
        //匿名类实现事件监听
        btn_red.addActionListener(new InnerClass());
        btn_blue.addActionListener(new InnerClass());
        this.getContentPane().add(btn_red);
        this.getContentPane().add(btn_blue);
        btn_blue.setBounds(new Rectangle(227, 115, 109, 45));
        btn_blue.setText("蓝色");
        this.setTitle("设置窗体背景色");
        this.setSize(300,300);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        Frame1 frame1 = new Frame1();
    }
    /*内部类**********************************/
    class InnerClass implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            Container c = getContentPane();
            if (e.getSource() == btn_blue) {
                c.setBackground(Color.blue);
            }
            if (e.getSource() == btn_red) {
                c.setBackground(Color.red);
            }
```

```
            }
        }
        /**************************************/
    }
```

2.7 实训

1．题目

设计一个多功能计算器，界面如图 2-13 所示。

图 2-13　计算器

2．要求

（1）实现加、减、乘、除运算。
（2）实现对非法数字的验证。
（3）实现平方根计算。
（4）实现倒数计算。
（5）【%】按钮对运算的结果进行除以 100。
（6）【Backspace】按钮是退格功能，输入有误时单击此按钮可以把输入错误的字符去掉。
（7）【C】按钮清除所记录的信息。
（8）【CE】按钮清除文本框中的信息。
（9）界面要美观大方，布局合理。

第 3 章　聊　天　室

3.1　项目目标

实现基于 C/S 模式的聊天室程序。
聊天室共分为服务器端和客户端两部分。

1．服务器端功能

（1）提供多线程的聊天服务处理。
（2）在服务器端屏幕上提示客户端进入，显示客户端聊天内容。
（3）当某客户端输入字符“quit”时断开该客户端的连接。

2．客户端功能

（1）可以配置要连接服务器的 IP 地址和端口号。
（2）配置用户昵称后连接。
（3）可以向所有人发送消息。
（4）显示本聊天室中的该用户登录后的所有消息。
（5）显示在该用户登录之后登录的用户登录信息。

3.2　项目分析

3.2.1　界面

服务器端界面只有一个文本区域，显示用户登录信息及全部在线用户所发送的消息，如图 3-1 所示。

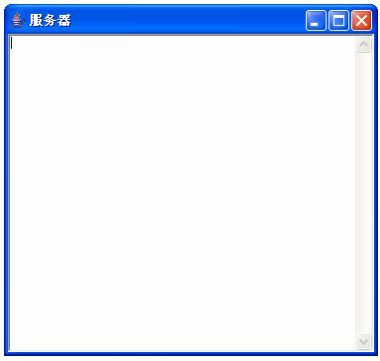

图 3-1　服务端界面

客户端界面包括如下部分：

（1）服务器 IP 设置。

（2）服务器端口设置。

（3）用户名设置。

（4）聊天记录显示。

（5）与服务器连接按钮。

（6）消息发送。

其界面如图 3-2 所示。

图 3-2　客户端界面

3.2.2　总体设计

1．服务器端设计

定义一个服务端类，实现服务器端功能。

（1）定义内部类，此内部类采用多线程，实现多客户端接入。

（2）当初始化界面时，启动服务器。

（3）当某客户端输入字符“quit”时断开该客户端的连接，关闭服务器窗口时关闭与客户端的连接。

（4）服务器接收客户端信息，并把这些信息发送到连接服务器的客户端。

2．客户端设计

定义一个客户端类，实现客户端功能。

（1）定义三个内部类，以实现连接按钮的监听、信息接收、对自己信息的监听。

（2）定义输入服务器的 IP 地址、端口号的控件和用户昵称，在连接时把这些信息传给服务器。

（3）发送消息时，通过对系统的监听显示自己发送的信息，其他用户发送的信息通过服务器转发。

3.3 代码思路及实现

3.3.1 代码思路

1．定义服务端类CServer继承Frame类

（1）定义两个构造方法，一个构造方法实现客户端不传参数时服务器窗口标题为空，另一个构造方法实现客户端传参数时服务器窗口标题为此参数值。都要执行窗口的初始化操作。

（2）init()方法即窗口初始化方法，实现服务端界面，在界面中定义一个文本区域，设定窗口关闭时系统退出，并启动服务器。

（3）boradCast(String str,Socket socket)方法，即向所有在服务器中的客户端广播方法，有两个参数，一个传递只显示的字符串，另一个传递套接字。该方法要判断发送信息的用户是否是自己。

（4）startServer()方法，启动服务方法。等待客户端的请求，接收到请求后，产生 socket 对象，交给多线程的聊天服务处理。

（5)定义内部类 Service 类,实现 Runnable 接口。有两个属性套接字 socket 和用户名 name。

① 定义两个构造方法，一个没有参数实现为空，另一个以套接字 socket 作参数，并获取用户名。

② run()方法，没有参数。当用户发送“quit”时，关闭客户连接。用户发送其他信息时广播信息。

（6）定义主方法，实例化 CServer 类。

2．定义CClient类继承Frame类

（1）把界面组件定义为属性，在 init()方法中构建界面，并调用与服务器连接的方法。

（2）connect()方法实现与服务器的连接。

（3）recive()方法实现接收服务器发送的信息。

（4）sendName()方法实现向服务器发送用户姓名。

（5）定义内部类 connBttnListener，实现 ActionLIstener。

（6）定义内部类 Recive，实现线程。

（7）定义内部类 MyListener，实现 ActionLIstener。

（8）定义 close()方法，实现关闭输出流，关闭套接字。

（9）定义主方法，实例化 CClient 类。

3.3.2 代码实现

服务器端程序：

```
import java.awt.Frame;
import java.awt.HeadlessException;
import java.awt.TextArea;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.BufferedReader;
```

```
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Enumeration;
import java.util.Hashtable;
//定义服务端类
public class CServer extends Frame {
TextArea textArea = new TextArea(20, 50);// 只有一个屏幕，提示客户端进入，以及打印客户端聊天内容
Socket socket =null;
Socket socket1=null;
Hashtable hashtable = new Hashtable();

public CServer() throws HeadlessException {
super();
init();
}
    public CServer(String arg0) throws HeadlessException {
        super(arg0);
        init();
    }
    // 画图，排列组件摆放位置
    public void init() {
        this.add(textArea);
        this.pack();
        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(-1);
            }
        });
        this.setVisible(true);
        // 启动 socket 服务
        startServer();
    }
    // 向所有在服务器中的客户端进行广播
    public void boradCast(String str,Socket self) {
        Enumeration enumeration = hashtable.keys();
        System.out.println("本聊天室共有"+hashtable.size()+"人");
        PrintStream printStream = null;
        textArea.append(str);
        while (enumeration.hasMoreElements()) {
            String s=(String) enumeration.nextElement();
            socket1= (Socket) hashtable.get(s);
            if(socket1!=self){
                try {
```

```
                    printStream = new PrintStream(socket1.getOutputStream());
                    printStream.println(str);
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
            }
        }
    }
    // 启动服务
    public void startServer() {
        try {
            ServerSocket server = new ServerSocket(3333);
            // 启动服务，一直等待客户端的请求
            while (true) {
                socket= server.accept();
                //接收到客户端的请求后，产生 socket 对象，交给多线程的聊天服务处理
                Service ser = new Service(socket);
                new Thread(ser).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
//定义内部类
    class Service implements Runnable {
        Socket socket = null;
        String name;
        public Service(Socket socket) {
            this.socket = socket;
            try {
  Buffered Readerbr1=new BufferedReader(new InputStreamReader(socket.getInputStream()));
                 name=br1.readLine();
                 hashtable.put(name,socket);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        public Service() {
        }
        public void run() {
            BufferedReader br = null;
            try {
                br = new BufferedReader(new InputStreamReader(socket
                        .getInputStream()));     // 得到输入流，可以得到客户输出流
                 boradCast(name+"进入聊天室\n",socket);
                // 保证不断地从客户端获得输入流，并输出到屏幕上
                while (true) {
```

```
                    String str = "";
                    System.out.println("服务器"+socket.isClosed());
                    if ((str = br.readLine()) != null) {
                        boradCast(name + "  说：" +
                                str + "\n",socket);
                    }
                    if ("quit".equals(str)) {
                        hashtable.remove(name);
                        break;
                    }
                }
                br.close();
                socket.close();
                textArea.append("关闭连接" + name);
            } catch (IOException e) {
                // e.printStackTrace();
            }
        }
    }
//定义主方法
    public static void main(String[] args) {
        CServer cs = new CServer("服务器");
    }
}
```

客户端程序：

```
import java.awt.BorderLayout;
import java.awt.Button;
import java.awt.Frame;
import java.awt.HeadlessException;
import java.awt.Label;
import java.awt.Panel;
import java.awt.TextArea;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.Socket;
import java.net.UnknownHostException;
//定义客户端类
public class CClient extends Frame {
```

```
TextArea textArea = new TextArea(18, 50);
TextField textField = new TextField(40);
TextField nickname = new TextField(20);
TextField address = new TextField("localhost", 10);
TextField port = new TextField("3333", 5);
Button sendBttn = new Button("发送");
Button connBttn = new Button("连接");
Panel panel1 = new Panel();
Panel panel2 = new Panel();
Socket socket = null;
PrintStream printStream = null;
public CClient() throws HeadlessException {
    super();
    init();
}
public CClient(String arg0) throws HeadlessException {
    super(arg0);
    init();
}
// 构建界面
public void init() {
    this.add(textArea);
    panel1.add(new Label("地址"));
    panel1.add(address);
    panel1.add(new Label("端口号"));
    panel1.add(port);
    panel1.add(new Label("昵称"));
    panel1.add(nickname);
    panel2.add(connBttn);
    panel2.add(textField);
    panel2.add(sendBttn);
    sendBttn.addActionListener(new MyListener());
    connBttn.addActionListener(new connBttnListener());
    this.add(panel1, BorderLayout.NORTH);
    this.add(panel2, BorderLayout.SOUTH);
    System.out.println(this.getTitle());
    this.pack();
    this.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            PrintStream printStream;
            try {
                printStream = new PrintStream(socket.getOutputStream());
                printStream.println("quit");
            } catch (IOException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
```

```
                close();
                System.exit(-1);
            }
        });
        this.setVisible(true);
        // connect();//链接服务器
    }
    public void connect() {
        try {
            socket = new Socket(address.getText(), Integer.parseInt(port.getText()));
            this.setTitle(nickname.getText());
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public void recive() {
        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(socket
                    .getInputStream()));
            String str = "";
            while ((str = br.readLine()) != null) {
                textArea.append(str + "\n");
            }
        } catch (IOException e) {
            System.exit(-1);
        }
    }
    public void sendName() {
        String name = nickname.getText();
        PrintStream printStream;
        try {
            printStream = new PrintStream(socket.getOutputStream());
            printStream.println(name);
        } catch (IOException e) {
            // e.printStackTrace();
        }
    }
    class connBttnListener implements ActionListener {
        public void actionPerformed(ActionEvent arg0) {
            // 链接服务器
            connect();
            // 发送个人昵称到服务器
            sendName();
            // 监听服务器消息
            Recive r = new Recive();
```

```
                new Thread(r).start();
            }
        }
        class Recive implements Runnable {
            public void run() {
                recive();
                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
        class MyListener implements ActionListener {
            public void actionPerformed(ActionEvent arg0) {
                try {
                    PrintStream printStream = new PrintStream(socket.getOutputStream());
                    printStream.println(textField.getText());
                    textArea.append("我说：" + textField.getText() + "\n");
                    textField.setText("");
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        public void close() {
            if (printStream != null)
                printStream.close();
            try {
                if (socket != null)
                    socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        public static void main(String[] args) {
            CClient cc = new CClient("客户端");
        }
    }
```

3.4 运行与发布

3.4.1 运行

1．服务器端程序运行

将 CServer.java 文件保存到一个文件夹中，如 e:\Chat\ChatServer。在使用 javac 命令进行编译之前，应使用如下命令设置类路径：

```
e:\Chat\ChatServer >set classpath= e:\Chat\ChatServer
```

然后利用 javac 命令对文件进行编译，使用如下命令：

```
javac CServer.java
```

之后，使用 java 执行程序：

```
java CServer
```

程序即运行服务器端程序界面。

2．客户端程序运行

将 CClient.java 文件保存到一个文件夹中，如 e:\Chat\ChatClient。在编译之前应设置路径，使用如下命令：

```
e:\Chat\ChatClient >set classpath= e:\Chat\ChatClient
```

然后利用 javac 命令对文件进行编译，使用如下命令：

```
javac CClient.java
```

之后，使用 java 执行程序：

```
java CClient
```

程序即运行客户端程序界面。

3.4.2 发布

1．服务器端程序发布

使用 jar.exe 将应用程序打包，把应用程序中涉及的类和图片压缩成一个 jar 文件，这样就可以发布程序了。

步骤如下：

（1）编写清单文件，名为 MANIFEST.MF，保存到 e:\Chat\ChatServer 文件夹下。其代码如下：

```
Manifest-Version: 1.0
Created-By: 1.5.0_02(Sun Microsystems Inc.)
Main-Class: CServer
```

（2）使用如下命令生成 jar 文件：

```
jar cfm CServer.jar MANIFEST.MF *.class
```

其中，c 表示要生成一个新的 jar 文件；f 表示要生成的 jar 文件的名字；m 表示清单文件的名字。

（3）为解决解压软件与.jar 文件的关联问题，在发布软件时还应该再编写一个 ChatServer.bat 文件。其中只有如下一条命令：

```
javaw –jar CServer.jar
```

以后可以通过双击 ChatServer.bat 来运行程序。

2．客户端程序发布

使用 jar.exe 将应用程序打包，把应用程序中涉及的类和图片压缩成一个 jar 文件，这样就可以发布程序了。

步骤如下：

（1）编写清单文件，名为 MANIFEST.MF，保存到 e:\Chat\ChatClient 文件夹下。其代码如下：

```
Manifest-Version: 1.0
Created-By: 1.5.0_02(Sun Microsystems Inc.)
Main-Class: CClient
```

（2）使用如下命令生成 jar 文件：

```
jar cfm CClient.jar MANIFEST.MF *.class
```

其中，c 表示要生成一个新的 jar 文件；f 表示要生成的 jar 文件的名字；m 表示清单文件的名字。

（3）为解决解压软件与.jar 文件的关联问题，在发布软件时还应该再编写一个 ChatClient.bat 文件。其中只有如下一条命令：

```
javaw –jar CClient.jar
```

以后可以通过双击 ChatClient.bat 来运行程序。

3.5 本项目实现中常见问题

使用多线程接收客户端信息的类要定义为内部类。这样，在接收时就可以调用广播方法 boradCast 来实现多用户端的广播。

3.6 项目技术支持

3.6.1 Java的输入/输出

1．输入/输出流概述

在实际应用中，程序在运行过程中要和外部进行信息交换，即向外部发送信息或从外部读取信息，这就是程序中所谓的输入/输出（I/O）操作。例如从文件中读取数据、向文件中写

入数据、从键盘中读取数据或向显示器显示数据等都是输入/输出操作。在 Java 程序中通过一种称为流的机制来实现输入/输出操作。而 Java 语言系统中的 java.io 包提供了标准的输入/输出流类库，通过该类库可以实现基本的输入/输出功能。

（1）输入/输出流的概念。流在 Java 语言中是指计算机中流动的数据缓冲区。一般分为输入流（Input Stream）和输出流（Output Stream）两类。从外部设备流向中央处理器的数据流称为“输入流”，反之称为“输出流”。在输入流中只能从输入流中读取数据，不能向输入流中写数据；在输出流中只能向输出流中写数据，而不能从输出流中读数据。常见的外部设备有打印机、显示器、键盘、扫描仪、硬盘、网络等。键盘只能做一个输入流，而显示器只能做一个输出流；但对于一个文件来说，当向其中写数据时，它就是一个输出流；当从其中读取数据时，它就是一个输入流。

java.io 包提供处理不同类型的流类，有字节流类、字符流类、文件流类和对象流类等。这些输入/输出流类都是从 java.io 包中的 InputStream、OutputStream、Reader 和 Writer 抽象类中继承而来的。Java 输入/输出类在 java.io 包中的层次结构如图 3-3 所示。其中 InputStream 类和 OutputStream 类是面向字节的输入/输出流类，而 Reader 类和 Writer 类是面向字符的输入/输出流类。File 类用于管理本地磁盘的文件和目录。RandomAccessFile 类用于文件的随机读/取操作。

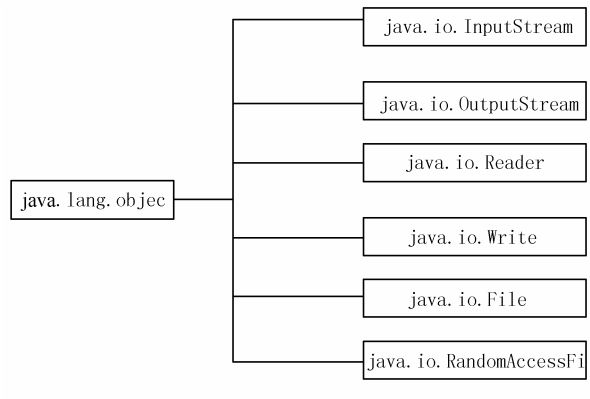

图 3-3　Java 输入输出类层次结构

（2）输入/输出流类。InputStream 和 OutputStream 类是 Java 中最基本的具有输入和输出功能的抽象类，也被称为基本输/入输出类。一些输入/输出流类都继承 InputStream 和 OutputStream 类，它们根据自身的特点又增加一些特殊的功能。

① InputStream 类。InputStream 抽象类中包含所有输入流需要的基本方法，用于从字节输入流中读取数据。Java 程序在从磁盘、键盘或其他计算机外部设备上读取数据时，首先需要创建一个输入流对象与外部设备连接，然后调用输入流对象的基本方法读取数据，数据读取完后再把输入流关闭。InputStream 类不能被实例化，只能通过子类来实现。InputStream 输入流类层次关系如图 3-4 所示。

StringBufferInputStream：此类用于从 StringBuffer 类中读取可变字符串数据。

ByteArrayInputStream：此类允许内存中一个缓冲区作为 InputStream 使用，通过将其同一个对象连接，可提供一个有用的接口。

FileInputStream：此类用于从本地文件系统中读取文件内容。

FilterInputStream：此类用于过滤 InputStream 输入流。

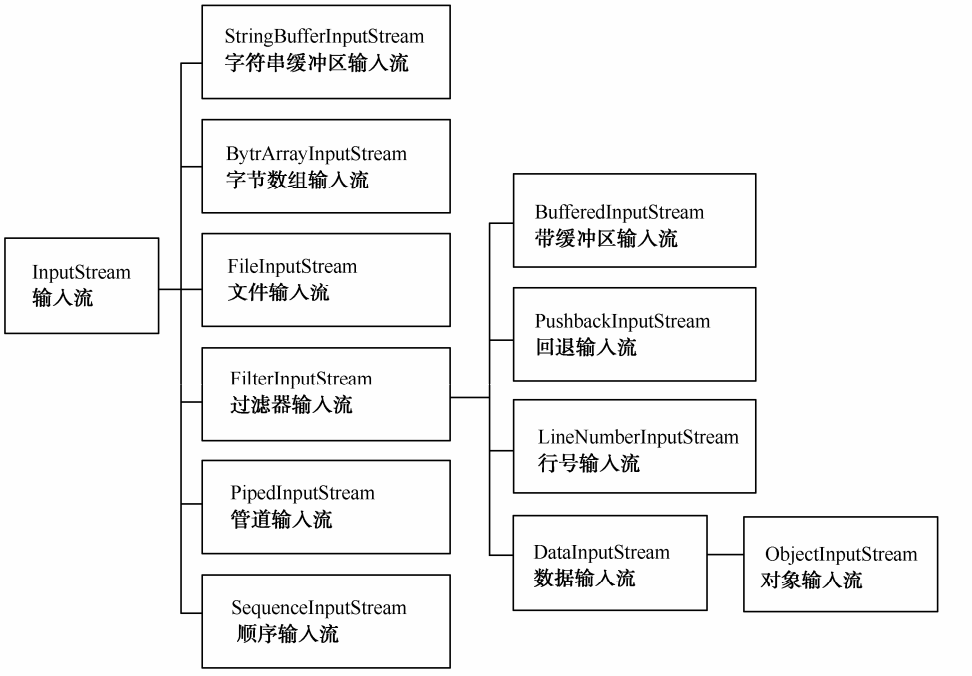

图 3-4 InputStream 子类

PipedInputStream：此类用于连接到传送输出流上，提供要写入传送输出流的所有数据字节。通常，数据由某个线程从 PipedInputStream 对象读取，并由其他线程将其写入到相应的 PipedOutputStream。

SequenceInputStream：此类建立顺序输入流。

BufferedInputStream：此类本身带有一个缓冲区，在读取数据时，先放到缓冲区中，可以减少对数据源的访问，提高运行效率。

PushBackInputStream：此类用于特殊的程序。实际上，读 PushBackInputStream 的程序从数据流读入多个字节，并形成包含多个字节组。有时为了确定一个组是否完整，程序必须读取第一个字符。一旦程序确定当前组是完整的，便向流中“写回”一个附加字符。一般用于编译器一类的程序中，用以解析输入的数据。

LineNumberInputStream：此类用于建立带有行计数功能的过滤输入流。

DataInputstream：此类提供一些基于多字节的读取方法，从而可以读取基本数据类型的数据。

ObjectInputStream：此类用于对对象的读取。

缓冲区是硬件设备间进行数据传输时，用来暂存数据的一个存储区域。它主要用于解决处理器与主存储器之间、处理器与其他外部设备之间和设备与设备之间通信速度的匹配问题。

InputStream 类提供的主要方法有以下几种。

public abstract int read( )：从输入流读取一个数据字节。返回一个 0～255 之间的整数。如果输入流当前位置没有数据，返回−1。

public int read(byte b[ ]) ：从输入流中读取一定数量的字节并将其存储在字节数组 b 中。

返回值是读取的字节个数。

public int read(byte b[ ], int off, int len)：从输入流中读取 len 个字节的数据，把数据存放在数组 b 中 off 下标开始处。

public int available( )：返回输入流中可以读取的字节数。注意：若输入阻塞，当前线程将被挂起。

public long skip(long n)：从输入流中跳过 *n* 个字节，返回是实际跳过的字节数。

public int close()：关闭此输入流，并释放与该流所有关联的系统资源。

② OutputStream 类。OutputStream 抽象类中包含所有字节输出流需要的基本写入方法。它与读操作一样，当 Java 程序需要向磁盘文件、显示器屏幕或其他计算机等外部设备输出数据时，首先建立一个输出流对象与外部设备连接，然后调用输出流对象的 write()方法将数据写到外部设备上，待写完数据后，关闭输入流对象。OutputStream 类是个抽象类，不能被实例化，只能通过子类实现。输出流类层次关系如图 3-5 所示。

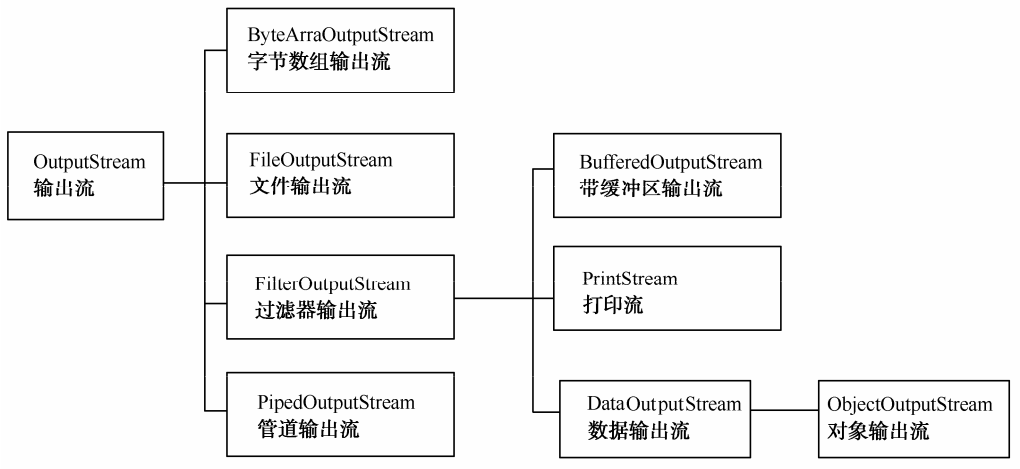

图 3-5 OutputStream 子类

ByteArrayOutputStream：此类实现了一个输出流，其中的数据被写入到一个字节数组中。

FileOutputStream：此类用于向本地文件系统中的文件写入数据。

FilterOutputStream：此类用于过滤输出流 OutputStream 类，使程序在写入数据时可进行数据处理。

PipedOutputStream：此类可以连接到传送输入流，以创建通信管道。传送输出流是管道的发送端。通常，数据由某个线程写入 PipedOutputStream 对象，并由其他线程从连接的 PipedInputStream 中读取。

BufferedOutputStream：此类是可实现缓冲的输出流。

PrintStream：此类为其他输出流添加了功能，使它们能够方便地实现打印各种类型数据。

DataOutputStream：此类允许应用程序以适当方式将 Java 基本数据类型写入输出流中。

ObjectOutputStream：此类将对象原始数据类型写入输出流中。

OutputStream 类提供的主要方法有以下几种。

public abstract void write(int b)：将整数 b 的低 8 位数据写到输出流中。

public void write(byte b[ ])：将数组 b 中的字节数据写到输出流中。

public void write(byte b[ ], int off, int len)：在数组 b 中，从数组下标 off 开始，读取 len 个字节数据，将这些数据写到输出流中。

public void flush()：将数据缓冲区中数据全部输出，并清空缓冲区。

public void close()：关闭输出流，并释放与输出流所有关联的系统资源。

（3）标准输入/输出。一般情况下，计算机都有默认的标准输入/输出设备。标准输入设备通常是指键盘，而标准输出设备是指显示器屏幕。从键盘获得数据或向显示器屏幕输出数据是经常遇见的操作。在实际应用中创建输入/输出流连接键盘和显示器屏幕很不方便，然而 Java 中提供代表键盘和显示器屏幕的对象分别是 System.in 和 System.out。

System 类是 java.lang 包中一个功能非常强大、非常有用的类，它提供了标准输入/输出以及运行时的系统信息。System 类是最终类，它的所有属性和方法都是静态的。in 和 out 是 System 类的两个静态的属性，分别代表系统的标准输入流和标准输出流。

① 标准输入流（System.in）。标准输入流 System.in 是 InputStream 类的对象。用户从键盘读入数据时，主要通过 System.in 的 read()方法来实现。read()有 3 种使用方式。

public int read()：参见 InputStream 类。

public int read(byte b[ ])：参见 InputStream 类。

public int read(byte b[ ], int off, int len)：参见 InputStream 类。

在执行 read()方法时从键盘缓冲区读取一个字节数据，得到的返回值是一个 int 数据。返回值中的高字节都是 0，低子节是读出的数据，是用户输入字符的编码，通过该编码可以知道输入了哪个字符。调用 read()方法会产生 IOException 异常，需要对该异常进行处理。

【例 3-1】 从键盘上读入一个字符，如果读入的是英文字符，则在屏幕上显示“是英文字符”，否则显示“是其他字符”。

```
import java.io.*;
public class IsEnglishChar{
    public static void main(String args[]){
        char c;
        int x;
        try{
            System.out.print("请输入一个字符：");
            x=System.in.read();                    //读取一个整型的数据
            c=(char)x;                             //将整型数据转换为字符数据
       //判断 c 字符变量是否为英文字母
             if((c>='a'&&c<='z')|| (c>'A'&&c<='Z'))
                System.out.println("是英文字符");
            else
                System.out.println("是其他字符");
       }catch(IOException e){
            System.out.println(e.getMessage());
       }
      }
}
```

程序运行结果如下：

```
C:\java>java IsEnglishChar
```

```
请输入一个字符：a
是英文字符
```

【例 3-2】　从键盘上读取一行数据，并显示在屏幕上。

```
import java.io.*;
public class ReadLineData{
    public static void main(String args[]) throws IOException{
        int count;                          //记录读入数据的字节数
        byte b[]=new byte[256];             //定义读入数据存放的缓冲区大小
        System.out.println("请输入一行数据，按回车结束");
        count=System.in.read(b);            //读入数据存放在 b 字节数组中
        //将字节数组中的数据转换为字符串数据，并去掉回车、换行字符
         String s=new String(b,0,count-2);
        System.out.println("输入的数据为："+s); }
}
```

程序运行结果如下：

```
C:\java>java ReadLineData
```

请输入一行数据，按回车结束
我喜欢 Java
输入的数据为：我喜欢 Java

② 标准输出流（System.out）。Java 的标准输出流 System.out 是打印输出流 PrintStream 类的对象。PrintStream 类又是过滤输出类 FilterOuptStream 的子类，它定义了向显示器屏幕输出不同类型数据的方法 pirnt()和 println()。print()方法是输出后不换行，而 println()方法是输出后换行。

【例 3-3】　print()方法和 println()方法的区别。

```
public class PrintDiffertoPrintln{
    public static void main(String arga[]){
      System.out.print("白日依山尽，"+"\n");
      System.out.print("黄河入海流。"+"\n");
      System.out.println("欲穷千里目，\n 更上一层楼。");    }
}
```

程序运行结果如下：

```
C:\java>java PrintDiffertoPrintln
白日依山尽，
黄河入海流。
欲穷千里目，
更上一层楼。
```

2．字节输入/输出流

字节输入/输出流提供对字节数据进行处理的输入/输出的基本方法，如在读写二进制数据时就要采用字节流。在 Java 语言中，字节输入/输出流类由 InputStream 和 OutputStream 两个抽象类派生而来，如图 3-4 和图 3-5 所示。

（1）字节输入流。字节输入流类很多，在这里主要介绍经常使用的 DataInputStream 类。

DataInputStream 类允许应用程序以与机器无关的方式从基本输入流中以字节的方式读取 Java 基本类型数据。它实现 DataInput 接口，DataInput 接口中定义了读取不同数据类型的各种抽象的读方法。

① 构造方法：DataInputStream(InputStream in)：创建一个字节流对象，并从基本输入流 in 对象中读取数据。

② 常用方法：

public int read(byte[] b)：参见 InputStream 类。

public int read(byte[] b, int off, int len)：参见 InputStream 类。

public boolean readBoolean()：读取布尔型数据。

public byte readByte()：读取字节型数据。

public char readChar()：读取字符型数据。

public double readDouble()：读取双精度型数据。

public float readFloat()：读取单精度型数据。

public int readInt()：读取整型数据。

public long readLong() ：读取长整型数据。

public short readShort()：读取短整型数据。

public int skipBytes(int n)：从输入流中跳过 *n* 个字节，返回是实际跳过的字节数。

【例 3-4】 利用 DataInputStream 类，接受键盘输入的数据，并显示在屏幕上。

```
import java.io.*;
public class GetKeyData{
   public static void main(String args[]) throws IOException{
      DataInputStream stdin=new DataInputStream(System.in);
      int count;                                    //记录读取字节总数量
      byte b[]=new byte[255];                       //定义缓冲区的大小
      System.out.print("请输入数据：");
      count=stdin.read(b);                          //读取的数据放在字节数组 b 中
      String s=new String(b,0,count-2);             //把字节数组中的数据转换为字符串
      System.out.println("输入的数据为："+s);
      stdin.close();                                //关闭输入流
}
}
```

程序运行结果如下：

```
C:\java>java GetKeyData
```

请输入数据：我喜欢 Java 语言

输入的数据为：我喜欢 Java 语言

（2）字节输出流。字节输出流类很多，在这里主要介绍 DataOutputStream 类。

DataOutputStream 类允许应用程序以适当的方式将 Java 基本数据类型以字节的方式写到输出流中。它实现 DataOutput 接口，DataOutput 接口中定义了写入不同数据类型的各种抽象的写方法。

① 构造方法：

DataOutputStream(OutputStream out)：创建一个对象，将数据写到基本输出流中。

② 常用方法：

public void flush()：参见 OutputStream 类。

public int size()：返回写到数据输出流中的字节数。

public void write(byte[] b, int off, int len)：参见 OutputStream 类。

public void write(int b)：参见 OutputStream 类。

public void writeChars(String s)：将字符串按字符顺序写到基本输出流。

public void writeDouble(double v)：使用 Double 类中的 doubleToLongBits 方法将 double 参数转换为一个 long 值，然后将该 long 值以 8 个字节的形式写入输出流中，先写入高字节。

public void writeFloat(float v)：使用 Float 类中的 floatToIntBits 方法将 float 参数转换为一个 int 值，然后将该 int 值以 4 个字节的形式写入输出流中，先写入高字节。

public void writeInt(int v)：将参数 v 以 4 个字节的形式写入输出流中，先写入高字节。

public void writeLong(long v)：将参数 v 以 8 个字节的形式写入输出流中，先写入高字节。

public void writeShort(int v)：将参数 v 以 2 个字节的形式写入输出流中，先写入高字节。

【例 3-5】 利用 DataOutputStream 类实现数据的输出。

```
import java.io.*;
public class OutputData{
    public static void main(String args[]) throws IOException{
        byte b[]=new byte[255]; //定义数据缓冲区
        int count;   //存放读入的字节数  //定义字节输入流
        DataInputStream stdin=new DataInputStream(System.in);
        //定义字节输出流
        DataOutputStream stdout=new DataOutputStream(System.out);
        System.out.println("请输入一行数据：");
        count=stdin.read(b); //从输入流中读取数据信息
        System.out.println("输入的数据为：");
        stdout.write(b,0,count); //把字节数组中的数据显示在屏幕上
        stdin.close();//关闭输入流
        stdout.close();//关闭输出流
}
}
```

程序运行结果如下：

```
C:\java>java OutputData
```

请输入一行数据：

我喜欢 Java！

输入的数据为：

我喜欢 Java！

（3）字节过滤流。过滤是指可以提供更多的功能，如缓冲、监视行号、或者将数据字节合成有意义的基本数据类型单元。在读/写数据的同时可以对数据进行处理，它提供了同步机制，使得某一时刻只有一个线程可以访问一个 I/O 流，以防止多个线程同时对一个 I/O 流进行操作所带来意想不到的结果。为了使用一个过滤流，必须首先把过滤流连接到某个输入/输出流上，通常通过在构造方法的参数中指定所要连接的输入/输出流。

常使用的过滤流有 BufferedInputStream、BufferedOutputStream、LineNumberInputStream、PushbackInputStream、PrintStream。这些类都是继承 FilterInputStream 和 FilterOutputStream 类。FilterInputStream 用于过滤 InputStream，而 FilterOutputStream 用于过滤 OutputStream。

在这里主要介绍 BufferedInputStream 和 BufferedOutputStream 类。

① BufferedInputStream 类。BufferedInputStream 类为输入流增添功能，即实现缓冲输入和支持 mark 和 reset 方法的能力。创建 BufferedInputStream 时即创建了一个带内部缓冲区数组。BufferedInputStream 类经常和其他的输入流类一起使用。

● 构造方法：

BufferedInputStream(InputStream in)：创建一个带有缓冲区字节过滤输入流对象，连接到基本输入流 in 对象上。

BufferedInputStream(InputStream in, int size)：创建一个指定大小的缓冲区字节过滤输入流对象，并连接到基本输入流 in 对象上。

● 常用方法：

public int available()：参见 InputStream 类。

public void close()：参见 InputStream 类。

public void mark(int readlimit)：在此输入流中标记当前的位置。

public int read()：参见 InputStream 类。

public int read(byte[] b, int off, int len)：参见 InputStream 类。

public void reset()：将重新定位到此输入流最后调用 mark 方法时的位置。

public long skip(long n)：参见 InputStream 类。

② BufferedOutputStream 类。该类实现缓冲的输出流。通过设置这种输出流，应用程序可以将各个字节写到基本输出流中，而不必为每次字节写入调用基础系统。

● 构造方法：

BufferedOutputStream(OutputStream out)：创建一个带有缓冲区的输出流对象，并将数据写入指定的基本输出流中。

BufferedOutputStream(OutputStream out, int size)：创建一个带有指定大小缓冲区的输出流对象，并将数据写入指定的基本输出流中。

● 常用方法：

public void flush()：刷新输出流。

public void write(byte[] b, int off, int len)：将字节数组 b 中数据，从数组下标 off 开始读取 len 个数据，写到输出流中。

public void write(int b)：将整型参数 b 写到输出流中。

【例 3-6】 利用 BufferedInputStream 类和 BufferedOutputStream 类实现数据的输入/输出。

```
import java.io.*;
public class InputOutputData{
    public static void main(String args[]) throws IOException{
        byte b[]=new byte[255];      //定义数据缓冲区
        int count;                   //定义读入的字节数
        //定义字节输入流
        DataInputStream stdin=new DataInputStream(System.in);
        //定义字节过滤输入流对象并与字节输入流对象关联
```

```
                BufferedInputStream bufin=new BufferedInputStream(stdin);
                //定义字节输出流
                DataOutputStream stdout=new DataOutputStream(System.out);
                //定义字节过滤输出流对象并与字节输出流对象关联
                BufferedOutputStream bufout=new BufferedOutputStream(stdout);
                System.out.println("请输入一行数据：");
                count=bufin.read(b); //从输入流中读取数据信息
                System.out.println("输入的数据为：");
                bufout.write(b,0,count); //输出字节数组中的数据
                bufin.close();     stdin.close();     //关闭输入流
                bufout.close();    stdout.close();    //关闭输出流
        }
        }
```

程序运行结果如下：

```
C:\java>java InputOutputData
```

请输入一行数据：
我喜欢 Java！
输入的数据为：
我喜欢 Java！

3．字符输入/输出流

在 Java 语言中字符是采用 Unicode 字符编码。汉字、英文字母和其他符号等都采用两个字节编码。在使用字节流处理汉字时，有时可能只读了汉字一个字节的编码，这种操作不当将会出现乱码，而采用字符流处理，就可能避免这种情况的发生。常见的字符输入/输出流是由 Reader 和 Writer 抽象类派生出来的。处理数据时是以字符为单位。

（1）字符输入流。字符输入流是由 Reader 抽象类派生出来的。Reader 类的子类如图 3-6 所示。

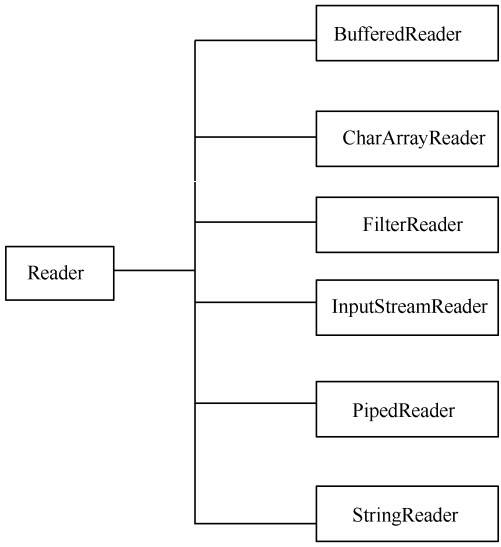

图 3-6　Reader 类的子类

① 常用字符输入流类：

BufferedReader：此类从字符输入流中读取文本。

CharArrayReader：此类实现一个可用做字符输入流的字符缓冲区。

FilterReader：此类用于读取已过滤的字符流的抽象类。

InputStreamReader：此类用于将字节翻译为字符的输入流。

PipedReader：管道字符输入流。

StringReader：读取字符串的字符输入流。

② 字符输入流类常用方法：

public int read()：从输入流中读取一个字符，返回值为读取字符的数量。

public int read(char buf[])：从输入流中读取字符存到数组 buf[]中，返回值为实际读取的字符的数量。

public int read(char buf[],int off,int len)：读取 len 个字符，从数组 buf 的下标 off 处开始存放，返回值为实际读取的字符数量。

public void mark(int readLimit)：给当前流做标记。

public void reset()：将当前流重置到做标记处。

public abstract void close()：关闭字符输入流并释放关联资源。

③ InputStreamReader 类。在字符输入流中主要介绍 InputStreamReader 类。InputStreamReader 类是将字节流转换为字符流的桥梁，它将读取的字节数据解释为字符编码。此类既可以指定字符集，也可以使用默认的字符集。InputStreamReader 类提供一个常用的构造方法：

InputStreamReader(InputStream in)：创建一个用默认字符集的字符输入流对象。

【例 3-7】 利用 InputStreamReader 类，实现直到输入"stop"一行字符串退出程序。

```
import java.io.*;
public class StopProgram{
public static void main(String args[])throws IOException{
   //定义字符输入流对象
   InputStreamReader stdin=new InputStreamReader(System.in);
   int count;                          //定义接受字符的个数
   char c[]=new char[255];             //定义存放数据缓冲区的大小
   String str;
   System.out.println("输入一行信息，输入 stop 退出程序");
   do {
      System.out.print("->");
      count=stdin.read(c);             //读取数据存放在数组 c 中
      str=new String(c,0,count-2);     //数据转换为字符串
    }while(!str.equals("stop"));
    stdin.close();                     //关闭输入流对象
   }
}
```

程序运行结果如下：

```
C:\java>java StopProgram
```

输入一行信息，输入 stop 退出程序：

```
->java
->buffer
->stop
C:\java>
```

（2）字符输出流

字符输出流是由 Writer 抽象类派生出来的。Writer 类的子类如图 3-7 所示。

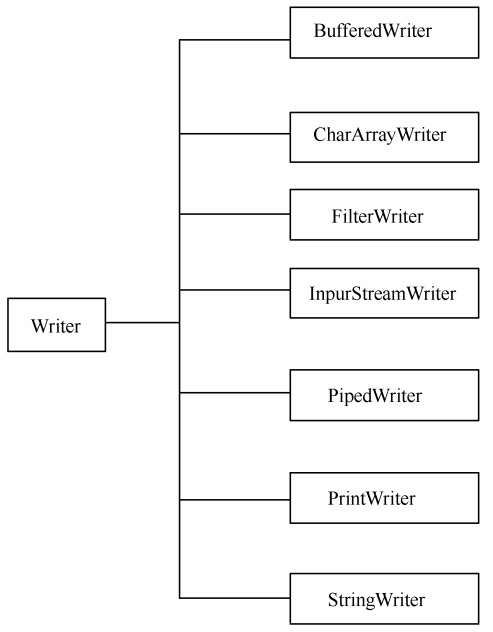

图 3-7 Writer 类的子类

① 常用字符输出流类：

BufferedWriter：此类将文本写入字符输出流中。

CharArrayWriter：此类实现一个可用做 Writer 的字符缓冲区。缓冲区会随输出流写入数据而自动增长。可使用 toCharArray()和 toString()检索数据。

FilterWriter：此类用于写入已过滤的字符流的抽象类。

OutputStreamWriter：此类是字节流转换为字符流的桥梁。它使用的字符集可以指定或采用平台默认的字符集。

PipedWriter：管道字符输出流。

PrintWriter：此类实现在 PrintStream 中的所有 print 方法。它不包含用于写入原始字节的方法，对于这些字节，程序应该使用未编码的字节流进行写入。

StringWriter：一个字符流，可以用其回收在字符串缓冲区中的输出来构造字符串。

② 常用方法：

public abstract void write(int b)：将参数 b 的低 16 位值写到输出流中。

public abstract void write(char buf[])：将字符数组 buf 中的值写到输出流中。

public abstract void write(char buf[],int off,int len)：将字符数组 buf 中的从下标为 off 位置

处开始的 len 个字符写到输出流中。

public abstract void write(String str)：将字符串 str 中的字符写到输出流中。

public abstract void flush()：输出所有缓存区中的数据，并清空输出流。

public abstract void close()：关闭输出流。

③ OutputStreamWriter 类。在字符输出流中主要介绍 OutputStreamWriter 类。OutputStreamWriter 类是将字节流转换为字符流的桥梁。可以实现各种数据的输出。OutputStreamWriter 类提供了一个常见的构造方法：

public OutputStreamWriter(OutputStream out)：创建一个字符输出流对象，并与 out 输出流对象关联。

【例 3-8】 利用 OutputStreamWriter 类实现各种数据的输出。

```
import java.io.*;
public class OutputChars{
   public static void main(String args[]) throws IOException{
       //定义字符输出流对象
       OutputStreamWriter stdout=new   OutputStreamWriter(System.out);
       char c[]={'H','E','L','L','O','\n'};
       stdout.write(65+"\n");              //输出整数
       stdout.write(3.1415+"\n");          //输出浮点数
       stdout.write("I like Java\n");      //输出字符串
       stdout.write('a'+"\n");             //输出字符
       stdout.write(c);                    //输出字符数组
       stdout.close();                     //关闭字符输出流
   }
}
```

程序运行结果如下：

```
C:\java>java OutputChars
65
3.1415
I like Java
a
HELLO
Press any key to continue...
```

（3）字符过滤流。

① BufferedReader 类。BufferedReader 类是从字符流中读取字符信息，将字符数据存在缓冲区内，可提供对单个字符、数组或行的方式高效读取。BufferedReader 类提供两个构造方法。

BufferedReader(Reader in)：创建一个使用默认大小输入缓冲区的字符输入流。

BufferedReader(Reader in, int size)：创建一个使用指定大小输入缓冲区的字符输入流。

为了提高效率，BufferedReader 和 InputStreamReader 类经常一起使用。BufferedReader 类提供了一个非常实用的方法 readLine()，该方法可以读取一行数据。其他方法可参见字符输入流类常用方法。

② BufferedWriter 类。BufferedWriter 类是带有缓冲区的字符流类，它提供对单个字符、

数组和字符串的高效写入。BufferedWriter 类提供两个构造方法。

BufferedWriter(Writer out)：创建一个使用默认大小输出缓冲区的字符输出流对象。一般情况下默认的缓冲区就足够大了。

BufferedWriter(Writer out, int size)：创建一个指定大小输出缓冲区的字符输出流对象。

BufferedWriter 类提供一个 newLine()方法，该方法写入一个行分隔符。行分隔符字符串由系统属性 line.separator 定义，并且不一定是单个新行('\n')符。其他方法可参见字符输出流类常用方法。

【例 3-9】 利用 BufferedReader 类和 BufferedWriter 类从键盘接收一行数据，直到输入的内容为一行"stop"字符串，则停止输入，并把输入的数据都打印在屏幕上。

```
import java.io.*;
public class NewStopProgram{
public static void main(String args[])throws IOException{
   //定义字符输入流对象
   InputStreamReader stdin=new InputStreamReader(System.in);
   //定义字符过滤流对象
   BufferedReader       bufin=new BufferedReader(stdin);
   //定义字符输出流对象
   OutputStreamWriter stdout=new OutputStreamWriter(System.out);
   //定义字符过滤流对象
   BufferedWriter         bufout=new BufferedWriter(stdout);
   String str;
   System.out.println("输入一行信息，输入 stop 退出程序");
   do{
      System.out.print("->");
      str=bufin.readLine();       //读取数据存放在 c 数组中
      bufout.write("输入的内容为："+str+"\n");
    }while(!str.equals("stop"));
      bufout.flush();              //清空缓冲区，并输出数据
      bufout.close();              //关闭输出流对象
      stdout.close();
      bufin.close();               //关闭输入流对象
      stdin.close();
   }
}
```

程序运行结果如下：

```
C:\java>java NewStopProgram
输入一行信息，输入 stop 退出程序
->I like Java
->VB
->Java
->sql2000
->我喜欢 Java
->stop
输入的内容为：I like Java
```

```
输入的内容为：VB
输入的内容为：Java
输入的内容为：sql2000
输入的内容为：我喜欢 Java
输入的内容为：stop
Press any key to continue...
```

4．文件输入/输出流

在实际应用中，程序员经常对文件进行操作，向文件中写入数据或从文件中读取数据。Java 语言提供了对文件进行读写操作的类。有 FileInputStream、FileOutputStream 字节文件输入/输出流类；FileReader、FileWriter 字符文件输入/输出流类；RandomAccessFile 随机访问文件类可以对文件的内容进行随机的读/写操作。FileInputStream 和 FileOutputStream 是 InputStream 和 OutputStream 的子类，FileReader 和 FileWriter 是 InputStreamReader 和 OutputStreamWriter 的子类，而 RandomAccessFile 是 Object 的子类。

（1）字节文件输入/输出流。字节文件输入/输出流在向文件中写数据或从文件中读数据的过程中采用字节方式处理。

① FileInputStream 类。FileInputStream 字节文件输入流类是在处理文件过程中以字节方式从文件中读取数据。

● 构造方法：

FileInputStream(File file)：创建一个指向 file 文件对象表示的文件中读取数据的文件输入流对象。

FileInputStream(String name)：创建一个指向文件名为 name 的文件中读取数据的文件输入流对象。

● 常用方法：

```
public int read()
public int read(byte[] b)
public int read(byte[] b, int off, int len)
public long skip(long n)
public void close()
public int available()
```

以上方法的解释参见字节输入流类。

② FileOutputStream 类。FileOutputStream 字节文件输出流类是以字节方式向文件中写入数据。

● 构造方法：

FileOutputStream(File file)：创建一个指向 file 文件对象表示的文件中写入数据的文件输出流对象。

FileOutputStream(File file, boolean append)：创建一个指向 file 文件对象表示的文件中写入数据的文件输出流对象。如果 append 参数为 true 则将数据写入文件末尾处，否则从文件开始处写入数据，将覆盖原数据。即 append 参数决定是覆盖数据还是追加数据。

FileOutputStream(String name)：创建一个由 name 指定的文件中，向其中写入数据的文件输出流对象。

FileOutputStream(String name, boolean append)：创建一个由 name 指定的文件中，向其中写入数据的文件输出流对象。如果 append 参数为 true 则将数据写入文件末尾处，否则从文件开始处写入数据。

- 常用方法：

```
public void close()
public void write(byte[] b)
public void write(byte[] b, int off, int len)
public void write(int b)
```

以上方法解释参见字节输出流类。

【例 3-10】 使用字节文件输入/输出流类，将文件的内容复制到一个空的文件中。

```
import java.io.*;
public class FileCopy1{
   public static void main(String args[]) throws IOException{
         FileInputStream    in=new FileInputStream("FileCopy1.java");       //创建文件输入流
         FileOutputStream out=new FileOutputStream("FileCopy1.txt");     //创建文件输出流
         int c;
         while(in.available()!=0)
{     //有可读数据就进行读操作，否则跳出循环
             c=in.read();       //从文件输入流中读取数据
             out.write(c);      //向文件输出流中写入数据
         }
         System.out.println("文件复制完毕");
         in.close();
         out.close();
      }
}
```

程序运行结果：

```
C:\java>java FileCopy1
```

文件复制完毕。

（2）字符文件输入/输出流。字符文件输入/输出流在从文件中读数据或向文件中写数据以字符方式来处理。Java 语言中 FileReader 和 FileWriter 两个类可以采用字符的方式对文件进行读写操作。

① FileReader 类。FileReader 字符文件输入流类是以字符方式从文件中读取数据。

FileReader 类构造方法如下：

FileReader(File file)：创建一个从 file 对象代表的文件中读取数据的文件输入流对象。

FileReader(String filename)：创建一个从 filename 对象指向的文件中读取数据的文件输入流对象。

FileReader 类常用方法继承了 InputStreamReader 类和 Reader 类的方法，可参见这两个类的常用方法。

② FileWriter 类。FileWriter 字符文件输出流类是以字符方式向文件中写入数据。

FileWriter 构造方法如下：

FileWriter(File file)：创建一个向 file 对象代表的文件中写入数据的文件输出流对象。

FileWriter(File file, boolean append)：创建一个向 file 对象代表的文件中写入数据的文件输出流对象。如果 append 参数为 true 则将字节写入文件末尾处，否则从文件开始处写入数据。

FileWriter(String filename)：创建一个向 filename 对象代表的文件中写入数据的文件输出流对象。

FileWriter(String filename, boolean append)：创建一个向 filename 对象代表的文件中写入数据的文件输出流对象。如果 append 参数为 true 则将数据写入文件末尾处，否则从文件开始处写入数据。

FileWriter 类常用方法继承了 OutputStreamWriter 类和 Writer 类的方法，可参见这两个类的常用方法。

【例 3-11】 利用字符文件输入/输出类，将文件的内容复制到一个空的文件中。

```
import java.io.*;
public class FileCopy2{
   public static void main(String args[]) throws IOException{
         FileReader    in=new FileReader("FileCopy2.java");//创建文件输入流
         FileWriter    out=new FileWriter("FileCopy2.txt");//创建文件输出流
         int c;
         while((c=in.read())!= -1){ //从文件输入流中读取数据
            out.write(c);                  //向文件输出流中写入数据
         }
         System.out.println("文件复制完毕");
         in.close();   //关闭输入流
         out.close(); //关闭输出流
      }
}
```

程序运行结果：

```
C:\java>java FileCopy2
```

文件复制完毕。

（3）文件的访问。File 类提供对操作系统目录的管理功能，主要用于文件命名、文件属性查询和处理文件目录、创建文件夹等操作。但是 File 类不能对文件内容进行读/写操作。File 类位于 java.io 包中。

① 构造方法：

File(File parent,String child)：根据 parent 抽象路径名和 child 路径名字符串创建一个 File 对象。

File(String pathname)：通过 pathname 将给定路径名创建一个 File 对象。

② 常用方法：

public boolean canRead()：判断文件是否可读。

public boolean canWrite()：判断文件是否可写。

public boolean delete()：删除指定路径下的文件或目录。

public boolean exists()：判断指定路径下的文件或目录是否存在。

public String getAbsolutePath()：获取绝对路径名称。

public String getName()：获取指定路径下的文件或目录的名称。
public boolean isFile()：判断指定路径下的文件是否是一个标准文件。
public boolean isHidden()：判断指定路径下的文件是否是一个隐藏文件。
public boolean mkdir()：在指定路径下创建一个目录。
public String[] list()：获取指定路径下的文件名和目录名。
public File[] listFiles()：获取指定路径下的文件名。
public long length( )：获得文件的长度，以字节为单位。
【例 3-12】　列出文件下所有的文件和子目录。

```
import java.io.*;
public class ReadFile{
   public static void main(String args[]) throws IOException{
         String filename;
         if(args.length>0)
            filename=args[0];
         else
            filename="c:\\java";
          File    in=new File(filename);
          String s[];
          s=in.list();
         System.out.println(filename+"文件及文件夹为：");
         for(int i=0;i<s.length;i++)
              System.out.println(s[i]);
       }
}
```

程序运行结果为：

```
C:\java>java ReadFile
c:\java 文件及文件夹为：
aaaa
FileCopy1.class
FileCopy1.java
FileCopy1.txt
FileCopy2.class
FileCopy2.java
FileCopy2.txt
GetKeyData.class
GetKeyData.java
IsEnglishChar.class
IsEnglishChar.java
Output.class
Output.java
OutputChars.class
OutputChars.java
PrintDiffertoPrintln.class
PrintDiffertoPrintln.java
```

```
ReadFile.class
ReadFile.java
ReadLineData.class
```

3.6.2 线程

1．多线程概述

（1）线程的概念。在编写线程程序之前，先了解与线程有关的几个基本概念。程序（program）是对数据描述与操作的一段静态代码的集合，是应用程序执行的依据。进程（process）是程序的一次动态执行过程，是系统运行程序的基本单位。进程从代码加载到执行完毕经历了从产生、发展到消亡的过程。线程是比进程更小的执行单位，是一个程序内部的顺序控制流。一个进程可以包含多个线程。Java 程序通过流控制来执行程序流，程序中单个顺序的流控制称为单线程。一个程序中可以同时运行多个不同的线程，执行不同的任务则把这些线程称为多线程。多线程意味着一个程序的多行语句在多处理器下可以同时运行，而在单处理器下分时运行，这种方式可提高程序的运行效率。

在同一个操作系统中，可以有多进程，多进程是同时运行多个任务，也就是同时运行多个程序。进程之间切换的开销比较大，每个进程都有自己独立的一块内存空间、一组系统资源，对每一个进程的内部数据和状态来说都是完全独立的。而对于线程来说，同一类线程共享一块内存空间和一组系统资源，但每个线程有独立的运行栈和程序计数器。线程区别于进程的就是，当多个线程之间进行切换时，系统负担远远小于进程。

程序、进程和线程的关系可以简单比喻为：winword.exe 文件（微软编写的 Word 软件的可执行文件）就是一个程序，打开 word 软件编写一份个人简历系统就执行了 winword.exe 程序的一个进程，而打印 Word 文档可以说执行了 Word 中的一个线程。

（2）Java 中的线程模型。随着计算机技术的发展，编程模型也越来越复杂多样化。但多线程编程模型是目前计算机系统架构的最终模型。随着 CPU 主频的不断攀升，X86 架构的硬件已经成为瓶颈，这种架构的 CPU 主频最高为 4G。事实上目前 3.6G 主频的 CPU 已经接近了顶峰。

如果不能从根本上更新当前 CPU 的架构（在很长一段时间内还不太可能），那么继续提高 CPU 性能的方法就是超线程 CPU 模式。作业系统、应用程序要发挥 CPU 的最大性能，就是要改变到以多线程编程模型为主的并行处理系统和并发式应用程序。所以，掌握多线程编程模型，不仅是目前提高应用性能的手段，更是下一代编程模型的核心思想。多线程编程的目的，就是“最大限度地利用 CPU 资源”，当某一线程的处理不需要占用 CPU 而只和外围设备等资源打交道时，让需要占用 CPU 资源的其他线程有机会获得 CPU 资源。从根本上说，这就是多线程编程的最终目的。

线程是程序中的一个执行流。一个执行流是由 CPU 运行程序的代码、程序涉及的数据所形成的。因此，线程被认为是以 CPU 为主体的行为。在 Java 中线程的模型就是一个 CPU、程序代码和数据的封装体。

Java 中的线程模型包含以下三部分：

① 一个虚拟的 CPU。该 CPU 封装在 java.lang.Thread 类中。此类是实现线程的基本类。

② 虚拟 CPU 执行的代码。代码是指线程要执行的具体动作，它可以与其他线程共享，

也可以不共享。当两个线程执行同一个类的实例代码时，就共享相同的代码。

③ 代码所操作的数据。数据与代码是独立的。数据也可被多个线程共享，当两个线程对同一个对象进行访问时，将共享数据。

（3）线程的状态和生命周期。Java 语言在默认情况下，每个 Java 程序只有一个默认的主线程，即只有一条执行路线。在 Application 程序中，主线程是 main()方法的执行顺序；对 Applet 程序，主线程是让浏览器加载并执行的 Java 小程序。要想设计多线程程序必须在主线程中创建新的线程对象。一个线程对象对应一个线程，也就只有一条执行路线，创建线程对象可以通过 Thread 类以及其子类来实现。对于每个线程都有一个从诞生到消亡的过程，常把这个过程称为线程的生命周期。线程的生命周期通常要经历 5 种状态，分别是新建状态、可运行状态、运行状态、不可运行状态、死亡状态，这 5 种状态之间的关系如图 3-8 所示。

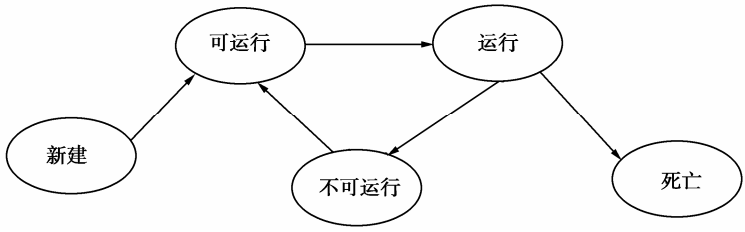

图 3-8　线程状态转换图

① 新建状态。新建状态是线程的第一个状态，通过 Thread 类或子类创建一个线程对象后，该对象就处于新建状态下。下面语句创建一个 newthread 的线程对象。

```
Thread    newthread=new Thread();
```

newthread 线程对象由于没有启动该线程，它处于新建状态，是一个空的线程，系统没有给它分配相应的内存空间和相关的资源。处于新建状态下的线程只能启动或终止，不能调用其他方法，否则就会失败并引起非法状态。

② 可运行状态。创建一个线程对象后，就可以调用 start()方法启动线程，这时的线程就处于可运行状态下。start()方法产生了运行这个线程所需的系统资源，并将线程加到线程队列中等待获得 CPU 资源。下面语句使线程处于可运行状态。

```
newthread.start();
```

由于大部分计算机只有一个 CPU，同一时刻只能运行一个处于可运行状态的线程，所以 Java 虚拟机建立一个线程队列，让这些线程以排队的方式轮流使用 CPU。当前线程运行结束时，就将队列中排在最前面的线程或优先级最高的线程从可运行状态转为运行状态。

③ 运行状态。当可运行状态的线程获得 CPU 资源时，即实际被 CPU 运行的线程，这时线程的状态被称为运行状态。线程进入运行状态就会自动调用该线程对象的 run()方法，run()方法定义该线程所要完成的操作和功能。

④ 不可运行状态。不可运行状态也称为阻塞状态。不可运行状态就是一个正在执行的线程在某种特殊情况下暂时中止运行。以下操作都会使线程进入不可运行状态。

- 执行费时的输入或输出操作。
- 调用 sleep()方法使线程进入睡眠状态。
- 调用 suspend()方法使线程进入挂起状态。

● 调用 wait()方法等待一个条件变量。

⑤ 死亡状态。死亡状态就是线程执行结束的状态，释放系统分配给它的资源。线程进入死亡状态有两种情况，分别为自然撤销和被停止。

自然撤销是指线程的 run()方法运行结束，退出系统的过程。被停止是指一个应用程序被停止运行。例如，用户从一个主页切换到另一个主页时，就会终止原主页中正在运行的所有线程。在 Java 中调用 stop()方法或 destroy()方法可以使线程进入死亡状态，前者会产生异常，后者是强制终止，不会释放锁。

⑥ 线程的调度和优先级。同一时刻如果有多个线程处于可运行状态，它们不可能同时获得 CPU 的使用权而运行，则它们需要排队等待 CPU 资源。此时多线程系统会给每个线程分配一个优先级标识，任务紧急的线程获得高优先级，反之则获得低优先级，优先级的高低也就反映了线程的重要或紧急程度。处于可运行状态的线程按优先级排队，优先级高的线程排在队列的前面，优先获得 CPU 的使用权；优先级较低的线程只能等待优先级高的线程运行结束后才能获得 CPU 的使用权；优先级相同的线程遵循“先进先出”的原则，即先进入可运行状态的线程优先有使用 CPU 的权利。

在 Java 中，提供了一个线程调度器来监控当前程序中进入可运行状态的所有线程。线程调度依据优先级基础上的“先到先服务”原则。线程调度管理器负责线程排队和 CPU 在线程间的分配管理，并由线程调度算法进行线程间的调度。线程调度器可以对不同的线程设置不同的优先级别，根据优先级高低选择线程执行，这样就可以让情况紧急的线程首先得到调度。线程的调度采用抢先式调度，即如果有一个比当前线程优先级更高的线程进入可运行状态，就会停止当前运行的线程，将当前线程加入等待队列中，而让高优先级的线程获得 CPU 的使用权，进入运行状态。但是如果在一个较低优先级线程的执行过程中，没有高优先级的线程进入可运行状态，那么低优先级的线程就会执行下去。抢先式调度可分为独占方式和分时方式。

独占方式是指当前活动线程一旦获得执行权，就会一直执行下去，直到执行完毕或由于优先权更高的线程处于可运行状态，或者是由于某种原因主动放弃 CPU，比如当前线程调用 sleep()方法等。

分时方式是指当前活动线程执行完当前时间片，如果有其他处于可运行状态的相同优先级的线程，系统就会将执行权转交给其他可运行状态的同优先级的线程，当前活动线程被加入等待执行队列的末尾，等待下一个时间片的调度。分时方式的调度系统使每个线程工作相同的时间，实现多线程同时运行。

2．线程的创建

在 Java 中实现线程可以通过直接方式创建线程或通过间接方式创建线程。直接方式是通过继承 Thread 类的子类来实现；而间接方式是通过 Runnable 接口和 Thread 类一起实现的。两种方式都要对 run()方法进行重写。

（1）Thread 类。Thread 类是创建线程的核心类，它代表 Java 程序中单个运行线程，综合了一个线程所需的属性和方法，java.lang 包中提供 Thread 类。下面介绍 Thread 类常用的方法和常量。

① 构造方法：

public　Thread()：创建一个新的线程对象。

public Thread(String name)：创建一个新的线程对象，并把 name 指定为其名称。

public Thread(Runnable target)：创建一个新的线程对象，并将 target 对象作为其运行对象。

public Thread(Runnable target,String name)：创建一个新的线程对象，并将 target 对象作为其运行对象，把 name 指定为其名称。

public Thread(ThreadGroup group,Runnable target)：创建一个新的线程对象，并将 target 对象作为其运行对象，同时把线程对象加到 group 线程组中。

public Thread(ThreadGroup group,String name)：创建一个新的线程对象，并把 name 指定为其名称，同时把线程对象加到 group 线程组中。

public Thread(ThreadGroup group,Runnable target,String name)：创建一个新的线程对象，并将 target 对象作为其运行对象，把 name 指定为其名称，同时把线程对象加到 group 线程组中。

② 静态方法：

public static Thread currentThread()：返回当前系统运行的线程对象。

public static int activeCount()：返回当前线程组中活动线程的个数。

public static void sleep(long millis)：让正在运行的线程休息 millis 毫秒后再运行，即暂停执行。

③ 常用方法：

public final String getName()：获得线程名字。

public final getPriority()：获得线程的优先级。

public final void setName(String name)：设置线程的名字为 name 所代表的内容。

public final setPriority(int priority)：设置线程的优先级。

public void start()：启动已创建的线程对象。如果获得 CPU 的使用权，就会自动调用 run()方法。

public final boolean isAlive()：判断线程是否在活动，如果是返回 true，否则返回 false。

public final ThreadGroup getThreadGroup()：获得当前线程所属的线程组名称。

public void run()：这是 Thread 线程类最重要的方法，是线程执行的起点，线程执行的具体操作都在此方法中进行编码。

public String toString()：获得线程的字符串信息。

④ 静态常量：在 Java 语言中，线程的优先级用 1～10 个数字表示，1 表示优先级最低，默认值是 5。Thread 线程类定义了三个静态常量。

```
public static final int NORM_PRIORITY=5
public static final int MIN_PRIORITY=1
public static final int MAX_PRIORITY=10
```

（2）Runnable 接口。如果用户希望由其他类派生的类可以支持多线程，那么必须在这个类中实现 Runnable 接口，因为 Java 中不允许类的多重继承。Thread 类本身就是实现 Runnable 接口。

Runnable 接口中只提供一个 run()方法，该方法与 Thread 类的 run()方法是一样的，在此方法中可编写控制线程的代码。任何实现 Runnable 接口的对象都可以作为一个线程 Thread 的目标对象。Runnable 接口位于 java.lang 包中。

（3）直接方式创建线程。简单创建一个线程，就是直接继承 Thread 线程类，在 Thread 子类中重写 run()方法，编写具体的操作代码。如果在程序中使用线程，只需要创建一个线程对象后，调用 start()方法即可。

采用这种方法实现线程的步骤如下：

① 定义 Thread 类的一个子类。

② 该子类中重写父类（Thread 类）中的 run()方法，编写具体的操作代码。

③ 创建该子类的一个线程对象。

④ 通过 start()方法启动线程，以便执行 run()方法。

【例 3-13】 线程技术演示。

```
//定义线程类
class Thread1 extends Thread{
String message;
Thread1(String name,String message){
      super(name);
      this.message=message;
}
//覆盖 run()方法，编写具体代码
   public void run(){
      for(int i=0;i<5;i++){
          System.out.println(this.getName()+"    "+message);
          try{
            //让线程休息 1 秒钟
              //sleep()方法产生异常，需要进行捕获
            Thread.sleep(1000);
          }
        catch(InterruptedException e){
            System.out.println(e.getMessage());
        }
      }
   }
}
//线程演示类
public class ThreadDemo1 {
public static void main(String args[]){
      //创建线程对象
      Thread1 t1=new Thread1("线程 1","打印数据 1");
      Thread1 t2=new Thread1("线程 2","打印数据 2");
      //启动线程对象
      t1.start();
      t2.start();
      }
}
```

程序运行结果如下：

```
线程 1   打印数据 1
```

```
线程 2　打印数据 2
线程 2　打印数据 2
线程 1　打印数据 1
线程 1　打印数据 1
线程 2　打印数据 2
线程 1　打印数据 1
线程 2　打印数据 2
线程 1　打印数据 1
线程 2　打印数据 2
Press any key to continue…
```

t1 和 t2 线程对象被线程调度管理器加到线程队列中，因为这两个线程的优先级相同，获得 CPU 的使用权的机会平等，所以运行有先有后。

【例 3-14】　线程之间的优先级演示。

```
//定义线程类
class Thread2 extends Thread{
String message;
Thread2(String message,int priority){
        this.message=message;
        //设置线程的优先级
        this.setPriority(priority);
}
      public void run() {
         for(int i=0;i<5;i++){
              try{
              sleep(1000);
             }catch(InterruptedException e){

       }
            System.out.println(message);
      }
   }
}
public class ThreadDemo2 {
   public static void main(String args[]){
      //创建线程对象，t1 线程对象优先级最高，t2 线程对象优先级最低
      Thread2 t1=new Thread2("线程 1 运行",Thread.MAX_PRIORITY);
      Thread2 t2=new Thread2("线程 2 运行",Thread.MIN_PRIORITY);
      //启动线程对象
      t1.start();
      t2.start();
      }
}
```

程序运行结果如下：

```
线程 1 运行
```

```
线程 2 运行
线程 1 运行
线程 2 运行
线程 1 运行
线程 2 运行
线程 1 运行
线程 2 运行
线程 1 运行
线程 2 运行
Press any key to continue...
```

线程 t1 和线程 t2 对象放入线程队列中，由于线程 t1 的优先级高于线程 t2 的优先级，所以每次线程 t1 运行结束后，线程 t2 才能获得 CPU 的使用权去运行。

（4）间接方式创建线程。由于 Java 语言不支持多重继承，用户想要使一个类实现线程可以通过继承 Runnalbe 接口，在该接口中有一个抽象的 run()方法，此方法与 Thread 类中的 run()方法一样。

采用这种方法实现线程的步骤如下：

① 定义一个实现 Runnable 接口的类。

② 实现 Runnable 接口中 run()方法，即在 run()方法中编写具体的操作代码。

③ 创建该类的一个线程对象，并将该对象作为参数，传递给 Thread 类的构造方法，从而生成 Thread 类的一个对象。

④ 通过 Thread 类生成的对象调用 start( )方法启动线程。

【例 3-15】 间接方式创建线程演示。

```
//定义线程类
class Thread3 implements Runnable{
String message;
String name;
Thread3(String name,String message){
      this.name=name;
      this.message=message;
}
//覆盖 run()方法，编写具体代码
  public void run(){
      for(int i=0;i<5;i++){
          System.out.println(name+"   "+message);
          try{
              //sleep()方法产生异常，需要进行捕获，让线程休息一秒钟
             Thread.sleep(1000);
          }
         catch(InterruptedException e){
             System.out.println(e.getMessage());
        }
     }
  }
}
```

```
public class ThreadDemo3 {
   public static void main(String args[]){
      //创建线程目标对象
      Thread3 d1=new Thread3("线程 1","打印数据 1");
      Thread3 d2=new Thread3("线程 2","打印数据 2");
      //创建线程对象
      Thread    t1=new Thread(d1);
      Thread    t2=new Thread(d2);
      //启动线程对象
      t1.start();
      t2.start();
      }
}
```

程序运行结果如下：

```
线程 1 运行
线程 2 运行
线程 1 运行
线程 2 运行
线程 1 运行
线程 2 运行
线程 1 运行
线程 2 运行
线程 1 运行
线程 2 运行
Press any key to continue...
```

3．多线程操作

在前面几节所提到的线程都是独立的，不是同步执行的，而是异步执行的，也就是说每个线程都包含自己运行时所需要的数据或方法，而不需要外部的资源，也不需要关心其他线程的状态或行为。一个 Java 程序的多线程之间可以共享数据。当线程以异步方式访问共享数据时，有时候是不安全的或者不合乎逻辑的。比如，同一时刻一个线程在读取数据，另外一个线程在处理数据，当处理数据的线程没有等到读取数据的线程读取完毕就去处理数据，必然得到错误的处理结果。这和前面提到的读取数据和处理数据并行多任务并不矛盾，这指的是处理数据的线程不能处理当前还没有读取结束的数据，但是可以处理其他的数据。比如，在网上看电影，下载线程负责从网络上下载数据，并把数据存放在本地缓冲区中；而播放线程负责从本地的缓冲区中读取数据，并播放。这时就要考虑下载线程和播放线程之间的状态关系和对缓冲区的操作结果，否则就会出现错误的播放结果。

（1）多线程的互斥。多个线程同时操作某一对象时，其中一个线程对该对象的操作可能会改变其状态，而该状态会影响另一线程对该对象的操作结果。像这种在某一时刻只能有一个线程执行某个执行单元的机制称做互斥控制或共享互斥。像上面网上看电影问题，不考虑下载线程和播放线程之间的状态关系，可能就会出现错误。下面用实例来模拟这种多线程之间互斥的关系。

【例 3-16】　模拟网上看电影，演示多线程的互斥关系。

```
//定义一个缓冲区的类
class Buffer{
    private int value;     //存放下载的数据
    //获得数据的方法
public    int getValue() {
          return value;
      }
    //存数据的方法
    public void putValue(int value) {
          this.value = value;
        }
   }
//定义播放线程
  class Play extends Thread {
        private Buffer buf;    //播放线程的缓冲区
       //定义一个构造方法
public Play(Buffer b) {
            buf=b;
        }
      //覆盖 run()方法
        public void run() {
              int value = 0;
              //从缓冲区中读数据
              for (int i = 1; i < 6; i++) {
                value = buf.getValue();
                System.out.println("用户播放数据片断为：" + value);
                try {
                    sleep((int)(Math.random() * 100)); //不定时的读入数据
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
              }
        }
}
//定义下载线程
class Down extends Thread {
    private Buffer buf;
    //定义构造方法
public Down(Buffer b) {
        buf = b;
    }
    public void run() {
        for (int i = 1; i < 6; i++) {
            buf.putValue(i);
            System.out.println("用户下载数据片断为：" + i);
            try {
```

```
                    sleep((int)(Math.random() * 100)); //不定时下载数据
                    } catch (InterruptedException e){
                        e.printStackTrace();
                    }
                }
            }
        }
//用户看电影
public class UserPlayDown{
        public static void main(String[] args) {
            Buffer buf = new Buffer(); //定义缓冲区对象
            Down d= new Down(buf); //定义下载线程对象
            Play p= new Play(buf);     //定义播放线程对象
            d.start();                 //启动下载线程
            p.start();                 //启动播放线程
        }
}
```

程序运行结果如下：

```
用户下载数据片断为：1
用户播放数据片断为：1
用户播放数据片断为：1
用户下载数据片断为：2
用户播放数据片断为：2
用户下载数据片断为：3
用户下载数据片断为：4
用户播放数据片断为：4
用户下载数据片断为：5
用户播放数据片断为：5
```

通过运行结果可以看出线程 Play 和线程 Down 不加以控制，就会出现下载线程下载数据存放到缓冲区后，而这时播放线程没有从缓冲区中读取数据，下载线程又开始下载数据存放到缓冲区中，这样就会把播放线程没有播放的数据覆盖；或者可能出现播放线程在播放完缓冲区的数据后，下载线程没有下载数据覆盖原先缓冲区中的数据，而播放线程又从缓冲区中读取数据开始播放，这样同一个数据就会播放两次。这两个线程就存在互斥关系，任何一个线程对数据的操作，都可能影响程序结果。

（2）多线程的同步。对于上例，不加以控制，就会出现错误。在 Java 中提供一种控制机制，当两个或两个以上的线程需要共享资源时，它们需要某种方法来确定资源在某一刻只能被一个线程占用，达到此目的的过程称为多线程同步。如果采用此方式，第一个线程在处理数据时，第二个线程不能访问该数据；当第一个线程处理完数据后，第二个线程才能访问数据；相反也是一样。这样就会避免错误。可见，线程同步是多线程编程的一个相当重要的技术。

Java 中对共享数据操作的并发控制是采用传统的封锁技术。为解决操作的不完整性问题，在 Java 语言中，引入了对象互斥锁的概念，来保证共享数据操作的完整性。每个对象都对应一个可称为“互斥锁”的标记，这个标记用来保证在任意时刻，只能有一个线程访问该对象。

用关键字 synchronized 来与对象的互斥锁联系。当某个对象用 synchronized 修饰时，表明该对象在任意时刻只能有一个线程访问。同时 Java 中还提供多线程之间通信的同步控制方法。

public final void wait()：使当前线程主动释放互斥锁，并进入该互斥锁的等待队列。

pubic final void notify()：唤醒 wait 队列中的第一个线程，并将该线程移入互斥锁申请队列。

public final void notifyAll()：唤醒 wait 队列中的所有线程，并将线程移入互斥锁申请队列。

【例 3-17】 模拟网上看电影，利用 synchronized 关键字、wait()和 notifyAll()方法，演示多线程的同步控制。

```
class Buffer{
    private int value;
    private boolean available=false;

    public synchronized int getValue() {
        while (available == false) {
            try {
                wait();// 等待下载线程写入数据
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            available = false;
            // 通知下载线程数据已经被取走，可以再次写入数据
            notifyAll();
        return value;
      }
    public synchronized void putValue(int value) {
            while (available == true) {
                try {
                    // 等待播放线程取走数据
                    wait();
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }
            }
            this.value = value;
            available = true;
           // 通知播放线程可以来取数据
            notifyAll();
        }
    }
 class Play extends Thread {
        private Buffer buf;
        private String name;
        public Play(Buffer b) {
            buf=b;
```

```
        }
        public void run() {
            int value = 0;
            for (int i = 1; i < 6; i++) {
                value = buf.getValue();
                System.out.println("用户播放数据片断为：" + value);
                try {
                    sleep((int)(Math.random() * 1000)); //采用随机时间播放下载数据
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
    class Down extends Thread {
        private Buffer buf;
        private String name;
        public Down(Buffer b) {
            buf = b;
        }
        public void run() {
            for (int i = 1; i < 6; i++) {
                buf.putValue(i);
                System.out.println("用户下载数据片断为：" + i);
                try {
                    sleep((int)(Math.random() * 1000)); //采用随机时间下载数据
                } catch (InterruptedException e){
                    e.printStackTrace();
                }
            }
        }
    }
    public class UserPlayDown1{
        public static void main(String[] args) {
            Buffer buf = new Buffer();
            Down d= new Down(buf);
            Play p= new Play(buf);
            d.start();
            p.start();
        }
    }
```

程序运行结果如下：

```
用户下载数据片断为：1
用户播放数据片断为：1
用户下载数据片断为：2
```

```
用户播放数据片断为：2
用户下载数据片断为：3
用户播放数据片断为：3
用户下载数据片断为：4
用户播放数据片断为：4
用户下载数据片断为：5
用户播放数据片断为：5
Press any key to continue...
```

通过采用相应的关键字和方法控制可以使多线程之间实现同步，从而保证了数据的一致性和正确性。

（3）线程的死锁。计算机系统中的各种资源都由操作系统进行管理和分配。线程所需要的资源可以在线程调度下来分配，根据线程执行情况可以动态地申请资源。当多个线程竞争共享资源时，可采用必要的措施来实现资源的互斥使用和保证线程能在有限的时间内获得所需的资源。但是这些管理和分配都是根据线程需求进行分配和调度的，而没有考虑一个线程已经占有的某些资源后又要申请其他资源时发生怎样的情况。当一个线程需要一个资源而另一个线程持有该资源不放时，就会发生死锁。例如，两个线程 t1 和 t2，它们都要使用资源 r1 和 r2，如果在分配资源时先把资源 r1 分给线程 r1，把资源 r2 分给线程 t2。线程 t1 和线程 t2 在分别得到了一个资源后都还想要得到第二个资源，在第二个资源没有得到前又不肯释放已经占有的资源。于是，线程 t1 和线程 t2 都只好等待对方释放占有的资源，然而两个线程占用对方所要的资源时又互不相让，就造成永远等待从而形成了死锁。

死锁是很难调试的错误，因为，通常情况下，它极少发生，只有当两个线程的时间段刚好符合时才能发生，它可能包含多于两个的线程和同步对象（也就是说，死锁在有更多复杂的事件序列的时候可能发生）。

Java 技术既不能发现死锁也不能避免死锁。所以程序员编程时应注意死锁问题，尽量避免。避免死锁的有效方法有：线程因为某个条件未满足而受阻，不能让其继续占有资源；如果有多个对象需要互斥访问，应确定线程获得锁的顺序，并保证整个程序以相反的顺序释放锁。

3.6.3 网络编程技术

1．网络基础

（1）网络。随着计算机应用的深入，特别是家用计算机越来越普及，一方面希望众多用户能共享信息资源，另一方面也希望各计算机之间能互相传递信息进行通信。个人计算机的硬件和软件配置一般都比较低，其功能也有限，因此，要求大型与巨型计算机的硬件和软件资源，以及它们所管理的信息资源应该为众多的微型计算机所共享，以便充分利用这些资源。基于这些原因，促使计算机向网络化发展，将分散的计算机连接成网，组成计算机网络。

计算机网络是现代通信技术与计算机技术相结合的产物。在计算机网络发展过程的不同阶段中，人们对计算机网络提出不同的定义，可分为广义的观点、资源共享的观点和用户透明性观点。从目前计算机网络发展的特点来看，资源共享观点的定义比较准确，能够准确描述出计算机网络的基本特征。计算机网络就是把分布在不同地理区域的计算机与专门的外部设备用通信线路互联形成一个规模大、功能强的网络系统，从而使众多的计算机可以方便地

互相传递信息，共享硬件、软件、数据信息等资源。通俗来说，网络就是通过电缆、电话线、或无线通信等互联的计算机的集合。

计算机网络可以实现以下三大基本功能：

① 计算机之间或计算机用户之间进行相互通信和交往。

② 共享资源，包括硬件资源、软件资源和数据与信息资源。

③ 计算机之间或计算机用户之间可以协同工作。

也就是说，通信、共享、协同工作是计算机网络三大基本功能。

网络上的计算机之间又是如何交换信息的呢？就像我们说话用某种语言一样，在网络上的各台计算机之间也有一种语言，这就是网络协议，不同的计算机之间必须使用相同的网络协议才能进行通信。所谓网络协议（Protocol）是一种特殊的软件，是计算机网络实现其功能的最基本机制。网络协议的本质是规则，即各种硬件和软件必须遵循的共同守则。网络协议也有很多种，具体选择哪一种协议则要看情况而定。Internet 上的计算机使用的是 TCP/IP 协议。常见的协议有 HTTP、FTP、Telnet、SMTP。

HTTP（Hyper Text Transport Protocol，超文本传输协议），是一个通用的、简单的，无状态、面向对象的协议，在 Internet 上进行信息传输时广泛使用。通过扩展请求命令，可以用来实现许多任务。HTTP 的允许系统相对独立于数据的传输，包括对该服务器上指定文件的浏览、下载、运行等。HTTP 不断发展，支持的媒体越来越多，如音频、视频、图像等数据，使得我们可以方便地访问 Internet 上的各种资源。

FTP（File Transfer Protocol，文件传输协议），是 TCP/IP 体系中的一个重要协议，它减少或消除在不同操作系统之间处理文件的不兼容性。即可以实现从一个系统向另一个系统传输文件。通过 FTP 用户可以方便地连接到远程服务器上，查看远程 FTP 服务器上的文件内容，还可以把所需要的内容复制到自己所使用的计算机上；如果 FTP 服务器允许用户对该 FTP 服务器上的文件进行管理，该用户就可以把自己计算机上的文件传送到文件服务器上，让其他用户共享，还能自由地对上面的文件进行编辑操作，如对文件进行删除、移动、复制、更名等。

Telnet（远程登录协议），提供了一个相当通用的、双向的、面向八位字节的通信机制，使用基于文本界面的命令连接并控制远程计算机。允许用户把自己的计算机当作远程主机上的一个终端，通过该协议用户可以登录到远程服务器上。Telnet 不仅允许用户登录到一个远程主机，还允许用户在那台计算机上执行命令。用户用 Telnet 登录到远程计算机上后，便可以通过自己本地的计算机来控制和管理远程服务器上的文件及其他资源。

SMTP（简单邮件传输协议），可以实现邮件传输可靠和高效。当用户给 SMTP 服务器发送请求时，一个双向的连接便建立起来，客户发一个 MAIL 指令，指示它想给 Internet 上某处的一个收件人发邮件。如果 SMTP 允许这个操作，一个肯定的确认发回客户机。随后，会话开始。客户可以告知收件人的名称和 IP 地址，以及要发送的消息。

（2）TCP/IP 协议。TCP/IP 协议（Transmission Control Protocol/Internet Protocol 传输控制协议/网际协议），是微软公司为了适应不断发展的网络，实现自己主流操作系统与其他系统间不同网络的互联而收购开发的，TCP/IP 协议是目前最常用的一种协议（包括 INTERNET），也可称得上是网络通信协议的一种通信标准协议，同时它也是最复杂、最为庞大的一种协议集。这些协议在功能上是不同的，范围很大，既包含像传输服务这样的普通任务，也包含提供扩展管理功能的复杂任务。TCP/IP 协议最早用于 UNIX 系统中，现在是 Internet 的基础协

议。作为互联网的基础协议，没有它就根本不可能上网，任何和互联网有关的操作都离不开 TCP/IP 协议。不过 TCP/IP 协议配置起来是最麻烦的一个，单机上网是比较简单的，而通过局域网访问互联网，就要给每个工作站分配一个 IP 地址、一个默认网关、一个子网掩码、一个主机等参数。TCP/IP 协议具有灵活性，可以支持任意规模的网络，几乎可以连接所有的服务器和工作站，正因为它的灵活性也带来了它的复杂性，同时它牺牲的是速度和效率。用户通常在安装操作系统时，不知不觉中就在自己的计算机上安装了 TCP/IP 协议包。

TCP/IP 体系共分为 4 个层次，如图 3-9 所示。

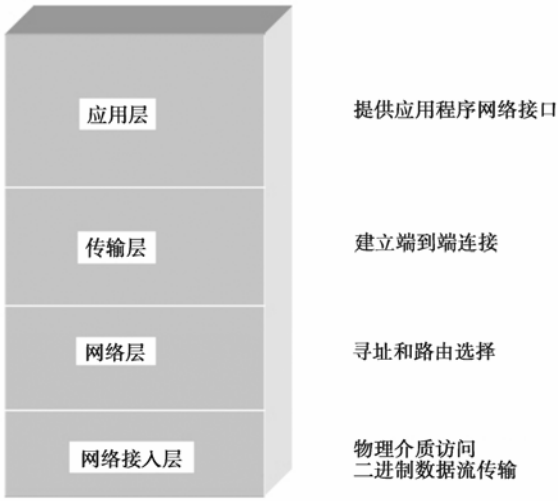

图 3-9　TCP/IP 体系层次

由于 TCP/IP 在设计时考虑到要与具体的物理传输媒体无关，因此在 TCP/IP 标准中并没有对数据链路层和物理层做出规定，而只是将最低的一层命名为网络接口层。这样如果不考虑没有多少内容的网络接口层，实际上只有三个层次。

TCP/IP 的最高层是应用层。许多著名的协议，如远程登录协议 TELENT 文件传输协议 FTP 简单邮件传送协议 SMNP 都在这一层。

下一层是传输层，它也叫主机到主机层，这一层可以使用两个不同的协议，一个是面向连接的传输控制协议 TCP（Transmission Control Protocol），另一个是面向无连接的用户数据报协议 UDP（User Datagram Protocol）。传输层传送的数据单位是报文或数据流。报文也常称为报文段。

网络层在传输层的下面，其主要的协议是无连接的网络协议 IP（Internet Protocol）。网络层传送的数据单位是数据报。与网际协议配合使用的还有 Internet 控制报文协议 ICMP，地址解析协议 ARP 和反向地址解析协议 RARP。

网络接入层，这是 TCP/IP 体系的最底层，负责接收 IP 数据包并通过网络发送，或者从物理网络上接受数据帧，抽出 IP 数据包，交给 IP 层。

（3）IP 地址和端口。

在 Internet 上有千百万台主机，为了区分这些主机，人们给每台主机都分配了一个专门的地址，称为 IP 地址。通过 IP 地址可以访问到 Internet 上的每一台主机。IP 地址由 32 位二进制数据组成。而通常 IP 地址中每 8 位二进制数字等效十进制数字表示，各部分之间用小数点分开。如某一台主机的 IP 地址为：202.98.83.68。IP 地址由 NIC（Internet Network Information Center）统一负责全球地址的规划、管理；同时由 Inter NIC、APNIC、RIPE 三大网络信息中

心具体负责美国及其他地区的 IP 地址分配。

为了方便对 IP 地址进行管理，Internet 的 IP 地址分为 5 类，如图 3-10 所示。通常 A 类、B 类和 C 类地址都由两个字段组成。

① 网络号字段。A 类、B 类和 C 类地址的网络号字段分别为 1、2 和 3 字节长，在网络号字段的最前面有 1～3 位的二进制数值分别规定 0、10 和 110。

② 主机号字段。A 类、B 类和 C 类地址的主机号字段分别为 3、2 和 1 字节长。

图 3-10　IP 地址分类

在网络技术中，端口大致有两种含义：一是物理意义上的真正端口。比如，集线器、交换机、路由器用于连接其他网络设备的接口，RJ-45 端口、SC 端口、光纤端口等。二是网络编程所使用逻辑意义上的端口，一般是指 TCP/IP 协议中的端口，比如用于浏览网页服务的 80 端口，用于 FTP 服务的 21 端口等。这里所指的端口就是逻辑意义上的端口，在网络服务中不同的端口将提供不同的服务，跟银行的窗口一样，不同的窗口提供不同的服务。

在 TCP/IP 协议中端口对应一个端口号，端口号是网络通信时同一机器上的不同进程的标识。TCP/IP 协议中的端口号是由一个 16 位的二进制组成，变化范围在 0～65535 之间。实际上，小于 1024 的端口号保留给预定义的服务，而且除非要和那些服务之一进行通信（例如 TELNETt、SMTP 邮件和 FTP 等），否则不能使用它们。两台计算机要实现互相之间的通信，就必须事先约定所使用的端口。如果使用的端口不一致，那么就不能实现互相通信。

（4）Socket 模式。在 20 世纪 80 年代的早期，美国加利福尼亚大学伯克利分校的一个研究组，开始研究如何将 TCP/IP 协议集成到 UNIX 系统内核中。在这个工作中，最重要的是要让 UNIX 用户进程与网络协议的交互作用顺利进行，在这个比传统 I/O 设备相互作用复杂得多的作用中，存在着两个问题。首先，如何让存在于不同主机上的两个网络操作进程建立它们的联系；其次，如何建立一种通用的机制以支持多种协议。这些其实都是网络编程界面所要解决的问题。

研究组很快解决这个问题，他们设计出一个接口，应用程序使用这个接口就可以方便地进行通信，这就是著名的“伯克利套接字”——Berkeley socket。随着 UNIX 系统的发展，Socket 接口得到了广泛的应用，如今已经广泛用到网络编程中，成为网络开发应用程序的强有力工具。

Socket 的英文含义为“插座”，它其实是指一个通信端点，借助于它，用户所开发的 Socket 应用程序可以通过网络与其他 Socket 应用程序进行通信，通信的基础就是 Socket。

2．URL类

在 java.net 包中的 URL 类是对统一资源定位符的抽象。使用 URL 类创建的应用程序称为客户端程序，一个 URL 对象存放着一个具体的资源的引用。一个 URL 对象包含最基本的 3 部分信息：协议、地址、资源。协议必须是 URL 对象所在的 Java 虚拟机支持的协议，有的协议将不支持，常见的 Http、Ftp 等协议是支持的。

（1）构造方法：

public URL (String url)：通过一个表示 URL 地址的字符串创建一个 URL 对象。

public URL(String protocol, String host, String file)：通过 protocol（协议）、host（主机）和 file（文件）3 个参数创建一个 URL 对象。

public URL(String protocol, String host, int port, String file)：通过 protocol（协议）、host（主机）、port（端口）和 file（文件）4 个参数创建一个 URL 对象。

通过 URL 类创建的对象将抛弃非运行时异常（MalformedURLException），因此必须要对这个异常进行处理。

（2）常用方法：

InputStream openStream()：返回一个指向 URL 对象所包含的资源的输入流。

String getHost()：返回 URL 对象中的主机信息。

int getPort()：返回 URL 对象的端口号。

String getProtocol()：返回 URL 对象的协议信息。

【例 3-18】　在文本框中输入 URL 地址，按 Enter 键将显示该网站的信息。

由于网络速度或其他的原因，URL 资源的读取可能会引起堵塞，程序将采用线程处理，避免堵塞。显示网页信息用 JEditorPane 组件。

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.swing.JEditorPane;
public class Demo implements ActionListener,Runnable{
    Frame f=new Frame();
    TextField t=new TextField(40);
    URL url;
    Thread thread=new Thread(this);
    JEditorPane    showhtml=new JEditorPane();
    byte b[]=new byte[255];
    public Demo(){
       showhtml.setEditable(false);
       ScrollPane scroll=new ScrollPane();
       f.add(t,BorderLayout.NORTH);
       t.addActionListener(this);
       scroll.add(showhtml);
       f.add(scroll,BorderLayout.CENTER);
       f.setSize(400,300);
       f.setVisible(true);    }
```

```
        public void run(){
            try{
                showhtml.setText(null);
                url=new URL(t.getText().trim());
                showhtml.setPage(url);
            }catch(IOException e){
                e.printStackTrace();
            }       }
        public void actionPerformed(ActionEvent ee){
             if(!(thread.isAlive()))
                 thread=new Thread(this);
              try{
                  thread.start();
              }catch(Exception e){
                   e.printStackTrace();
              }          }
        public static void main(String args[]){
            new Demo();
    }     }
```

程序运行的结果如图 3-11 所示。

图 3-11　URL 实例运行结果

3．TCP Socket通信

（1）TCP 客户端。在 Java 语言中创建 TCP 客户端程序通常需要 4 步。

① 需要创建一个 Socket 对象连接到服务器上。Socket 类提供几个构造方法，用于创建不同的 Socket 对象。

```
        public Socket(String host , int port)
        public Socket(InetAddress address, String host ,   int port)
```

其中，参数 host 代表要连接服务器的主机名，参数 address 代表要连接服务器的 IP 地址，参数 port 代表服务器提供给客户端服务的端口号。

```
        Socket   client=new   Socket("ServerName",9090);
```

上面语句创建一个 Socket 对象 client，client 将与名称为“ServerName”的服务器连接，该服

务器提供的通信端口为 9090。同时通过 client 对象向服务器发出请求，以便于服务器建立通信连接。

② 利用 Socket 类提供的 getInputStream()方法或 getOutputStream()方法，来获得服务器发来的信息或向服务器发送信息。

```
InputStream input=client.getInputStream();          //获得服务器发送的信息
OutputStream output=client.getOutputStream();       //向服务器发送的信息
```

③ 处理输入流或处理输出流，从输入流中读取数据可调用 read()方法，向输出流中写入数据可以调用 write()方法。

④ 关闭输入流或输出流、Socket 对象。当通信结束时，可以调用 Socket 对象的 close()方法关闭 Socket 对象。

（2）TCP 服务器。在 Java 语言中创建 TCP 服务器程序通常需要 5 步。

① 需要创建一个 ServerSocket 对象，以便和客户端进行通信。ServerSocket 类提供几个构造方法，用于创建不同的对象。

```
public ServerSocket(int port)
public ServerSocket(int port,int maxcount)
public ServerSocket(int port,int, maxcount, InetAddress ip)
```

其中，参数 port 代表服务器提供服务的端口号，参数 maxcount 代表服务器最多可以接受连接到服务器的客户端数量。

```
ServerSocket server=new ServerSocket(9090);
```

上面语句创建一个 ServerSocket 对象，进行通信的端口号为 9090。

② ServerSocket 对象与客户端建立通信连接。ServerSocket 对象调用 accept()方法等待客户端的服务请求，当接收到客户的服务请求时，accept()方法返回一个 Socket 对象，该对象包含客户端的基本信息。可以通过下面的语句实现。

```
Socket     connect =serve. accept();
```

③ 与客户端进行通信，调用 ServerSocket 对象的 getInputStream()方法或 getOutputStream()方法，来获得客户端发来的信息或向客户端发送信息。

```
InputStream input=client.getInputStream();          //获得服务器发送的信息
OutputStream output=client.getOutputStream();       //向服务器发送的信息
```

④ 处理输入流或处理输出流，从输入流中读取数据可调用 read()方法，向输出流中写入数据可以调用 write ()方法。

⑤ 关闭输入流或输出流及 SocketServer 对象。当通信结束时，可以调用 Socket 对象的 close()方法关闭 SocketServer 对象。

【例 3-19】 创建一个简单的服务器和客户端通信程序。

客户端程序：

```
import java.net.*;
import java.io.*;
class TestClient {
   public static void main(String args[]) {
      try {
```

```
            Socket client = new Socket("192.168.1.1",9090);
            InputStream input = client.getInputStream();
            DataInputStream in = new DataInputStream(input);
            byte buf[]=new byte[255];
              int n;
              n=in.read(buf);
              String s=new String(buf,0,n);
              System.out.println(s);
            in.close();
              input.close();
            client.close();
            } catch (ConnectException e1) {
                System.err.println("服务器连接失败！");
          } catch (IOException e2) {}
      }
  }
```

服务器端程序：

```
  import java.net.*;
  import java.io.*;
  class TestServer {
     public static void main(String args[]) {
       String s= "Hello";
       try {
            ServerSocket server = new ServerSocket (9090);
              Socket    connect=server.accept();
            OutputStream output = connect.getOutputStream();
            DataOutputStream out = new DataOutputStream(output);
              out.writeUTF(s);
            out.close();
              output.close();
            server.close();
         } catch (ConnectException e1) {
         } catch (IOException e2) {}
      }
  }
```

4．UDP Socket通信

（1）UDPSocket通信概述。UDP进行通信从整体上看可分成两大步骤，第一步是发送数据报，第二步是接收数据报。采用这种方式编写的客户端/服务端程序时，无论在客户端还是服务端，首先都要建立一个DatagramPacket对象，封装要发送的数据、数据报分组长度、发送目的地主机的IP地址和端口号或封装接收数据报的缓冲区和数据报的长度。然后使用DatagramSocket对象，用来接收或发送数据报。

① 发送数据报。发送数据报需要做 3 步工作。

- 创建一个 DatagramPacket 对象。可通过下面的构造方法实现。

```
DatagramPacket（byte buf[],int length）
DatagramPacket(byte buf[], int length, InetAddress address, int port)
DatagramPacket(byte[] buf, int offset, int length)
DatagramPacket(byte[] buf, int offset, int length, InetAddress address, int port)
```

其中，buf 中存放数据报数据，length 为数据报中数据的长度，address 和 port 指明目的地址和端口号，offset 指明了数据报的位移量。

创建一个数据报对象并封装相关信息，可以通过下面的语句实现。

```
byte     data[]=new byte[100];
DatagramPacketsendpacket=newDatagramPacket(data,data.length,InetAddress.getLocalHost(),8888);
```

InetAddress.getLocalHost()代表目标机器的 IP 地址，8888 代表与目标机器通信的端口。

- 创建一个 DatagramSocket 对象，绑定 IP 地址或端口可通过下面几个构造方法实现。

```
DatagramSocket()
DatagramSocket(int prot)
DatagramSocket(int port, InetAddress ip)
```

其中，port 指明 socket 所使用的端口号，如果未指明端口号，则把 socket 连接到本地主机上一个可用的端口。ip 指明一个可用的本地地址。给出端口号时要保证不发生端口冲突，否则会生成 SocketException 异常。上述的两个构造方法都声明抛弃非运行时异常 SocketException，程序中必须进行处理，或者捕获、或者声明抛弃。通过下面语句创建一个 DatagramSocket 对象。

```
DatagramSocket    socket=new DatagramSocket();
```

- 调用 DatagramSocket 对象的 send()方法发送数据报。send()方法以 DatagramPacket 对象为参数。此方法使用时将产生 IOException 异常。通过下面语句实现发送数据报。

```
Socket. send(sendpacket);
```

② 接收数据报。接收数据报则需要做下面 4 步工作。

- 首先创建一个 DatagramPacket 对象，用来接收数据。采用 DatagramPacket（byte buf[],int length）构造方法创建对象，该方法指明接收数据的缓冲区及长度。可以通过下面的语句实现。

```
byte     data[]=new byte[100];
DatagramPacket receivepacket=new DatagramPacket(data,data.length);
```

- 创建一个 DatagramSocket 对象，用来绑定目标的 IP 地址和端口号。通过下面的语句可以实现。

```
DatagramSocket    socket=new DatagramSocket(8888，InetAddress.getLocalHost());
```

- 调用 DatagramSocket 对象的 receive()方法接收数据。receive()方法中的参数为 DatagramPacket 对象，该方法使用时将产生 IOException 异常。receive()方法接收数据时将一直等待，直到收到一个数据报存入到 DatagramPacket 对象为止。可通过下面的语句来实现。

```
socket. receive(receivepacket);
```

● 处理接收的数据。调用 DatagramPacket 对象的 getData()来获得数据。

（2）UDP 通信实例。

【例 3-20】 下面是一个简单的 UDP 通信实例。编写一个服务端程序和客户端程序，客户端程序向服务端程序发送一个字符串。

客户端程序：

```
import java.net.*;
import java.io.*;
class TestClient1 {
   public static void main(String args[]) throws BindException {
       String message="hello";
       byte data[]=message.getBytes();   //定义发送的数据
       try {
           //创建 DatagramPacket 对象
            DatagramPacket sendpacket=new DatagramPacket(
data,data.length,InetAddress.getLocalHost(),8888);
            //创建 DatagramSocket 对象
DatagramSocket client=new DatagramSocket();
            //发送数据
client.send(sendpacket);
}catch (IOException e) {
              System.out.println("程序运行出错:" + e);
       }
    }
}
```

服务器端程序：

```
import java.net.*;
import java.io.*;
class TestServer1{
       public static void main(String args[]) {
          byte buf[]=new byte[100];
          try {
               //创建 DatagramPacket 对象
DatagramPacket receivepacket=new DatagramPacket(buf,buf.length);
//创建 DatagramSocket 对象
                DatagramSocket server=new DatagramSocket(8888,InetAddress.getLocalHost());
                while(true){
                 //创建数据
                    server.receive(receivepacket);
                //处理接受的数据
                 String
s=new String(receivepacket.getData(),0,receivepacket.getLength());
                 System.out.println(s);
              }
```

```
                }catch (IOException e2) {}
            }
        }
```

3.7 实训

1．题目

完善聊天室程序，服务器界面如图 3-12 所示，客户端界面如图 3-13 所示。

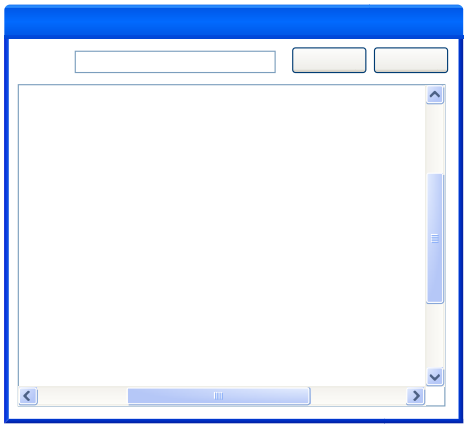

图 3-12　聊天室服务器界面

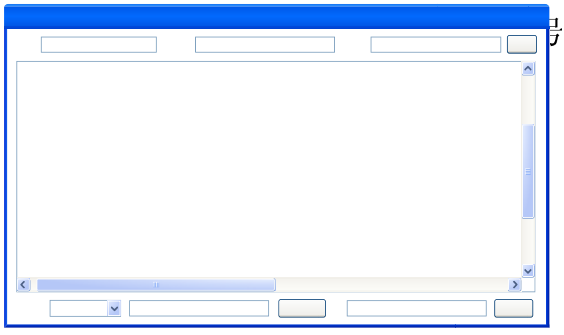

图 3-13　聊天室客户端界面

2．要求

（1）服务器端功能：

① 服务器端添加端口设置功能。

② 添加启动服务、停止服务功能。

（2）客户端功能：

① 添加选择发送对象功能。

② 添加发送文件功能。

第 4 章　二十一点游戏

4.1　项目目标

利用 Java swing 技术能够实现玩家与电脑进行二十一点游戏。

要求如下：

纸牌数：共 52 张纸牌，除去大小王两张纸牌。

花色：红桃、黑桃、方块、梅花。

纸牌的面值：A 到 10 的纸牌面值按照 1 到 10 计算，J、Q、K 的纸牌面值按照 1 进行计算。

玩法：电脑先抓牌，玩家后抓牌。计算自己的面值总数，比较面值数；如果面值总数都大于或等于 21，则平局；如果玩家和电脑的面值总数有一个大于 21 点，另一个不大于 21 点，则不大于 21 点的为赢家；如果玩家和电脑的面值总数都不大于 21，则面值总数大的为赢家。

游戏运行过程如下：

（1）程序运行开始界面，如图 4-1 所示。

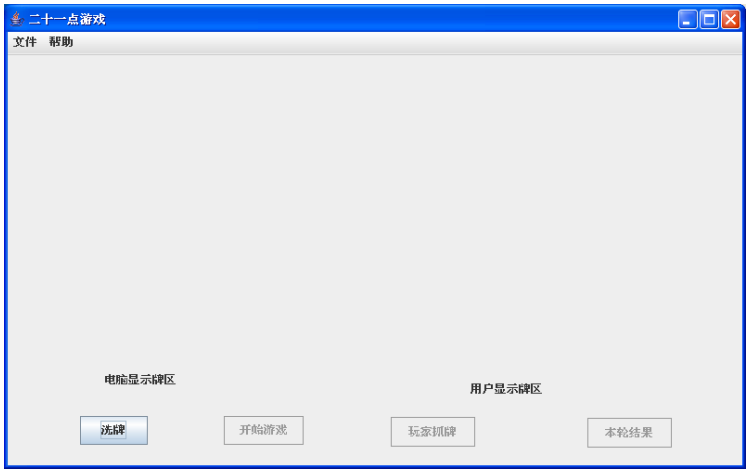

图 4-1　游戏开始界面

（2）单击【洗牌】按钮后，界面如图 4-2 所示。

（3）单击【开始游戏】按钮后，界面如图 4-3 所示。

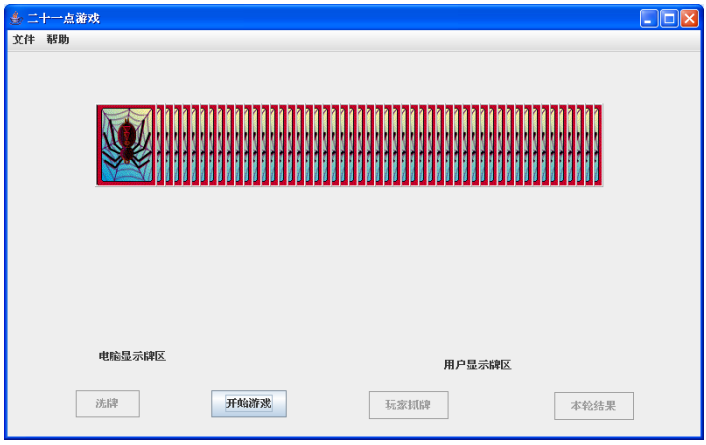

图 4-2　游戏洗牌界面

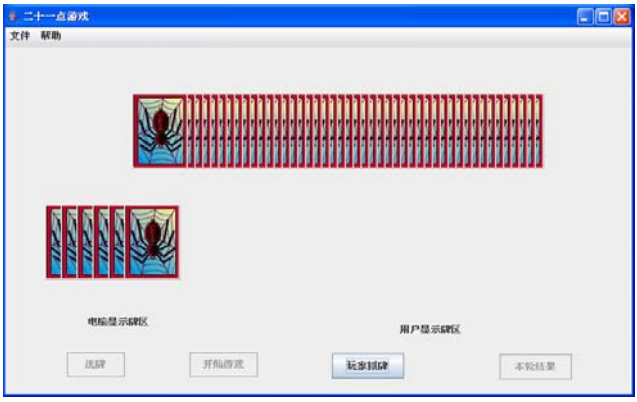

图 4-3　电脑抓牌界面

（4）单击【玩家抓牌】按钮后，界面如图 4-4 所示。
（5）单击【本轮结果】按钮后，界面如图 4-5 所示。
（6）单击【文件】下的【退出】菜单，程序结束运行。
（7）单击【帮助】下的【关于】菜单，弹出如图 4-6 所示窗体。

图 4-4　玩家抓牌界面

图 4-5　游戏结束

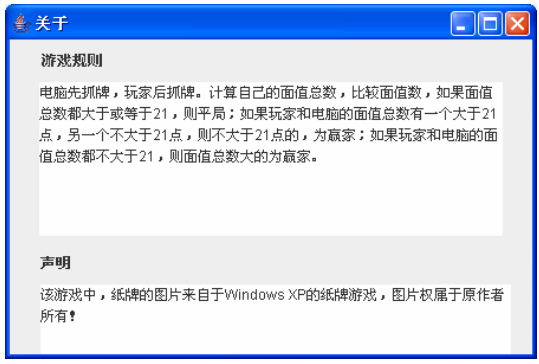

图 4-6　【关于】窗体

4.2　项目分析

1．算法

（1）生成一副符合要求的纸牌，以纸牌的背面显示给玩家，即洗牌。

（2）游戏开始，电脑先依次抓牌，抓牌的总面值数大于等于 15 点就停止抓牌。并以牌的背面显示给玩家。

（3）玩家依次抓牌，玩家根据牌的总面值确定是否再抓牌。以牌的正面显示给玩家。

（4）玩家确定不再抓牌，即本次游戏结束，计算胜负关系，显示玩家和电脑的胜负关系。

2．牌的图片命名规则

在游戏中的纸牌以图片的方式显示，因此对图片的命名与纸牌的面值和花色进行关联。规则如下。

（1）“1”代表红桃，“2”代表黑桃，“3”代表方块，“4”代表梅花。

（2）“1 到 10”代表面值为“A 到 10”，11 代表面值为 J，12 代表面值为 Q，13 代表面值为 K。

（3）图片的名称由花色、连字符“-”和面值三部分信息组成，如 1-1.gif 代表红桃 A，4-11.gif 代表梅花 J。

3．一张牌的设计

根据牌的特性：设计一个 Card 类，含有两个属性 type 和 value，分别表示牌的花色和面值。创建一个对象就等于生成一张牌。

4．一副牌的设计

定义 Card 数组，长度为 52，把 52 张牌放到数组中。要实现随机放置牌到数组中，首先生成一副有规律的牌，依次生成红桃 A 到红桃 K、黑桃 A 到黑桃 K、方块 A 到方块 K、梅花 A 到梅花 K，共 52 张牌，同时确定花色和面值，依次放入 Card 数组中。再循环 52 次，每次生成两个 0～51 之间的整数，依据利用这两个随机数交换 Card 数组中对应位置的内容，即随机打乱这幅牌。

5．类的设计

对本项目实现的功能设计了 4 个类：Card、CardManager、GameFrame 和 AboutFrame。

Card 类主要是对一张纸牌的设计；CardManager 类主要是随机生成一副 52 张纸牌，并在容器中发牌；GameFrame 类主要负责游戏界面和玩法等；AboutFrame 类主要显示游戏规则和纸牌图片的版权信息。

CardManager 类中定义了 Card 数组，长度为 52，用于存放纸牌。主要方法如下：

（1）void initCards()按照一定规则初始化一副 52 张纸牌。

（2）void randomCards()随机打乱这副 52 张纸牌。

（3）void gameStart(JLabel game[],Container c)用于在容器中显示纸牌的图片，其中参数 game[]是标签框控件数组，用于显示图片；参数 c 代表显示纸牌的容器，即 Frame 对象。

GameManager 类实现了 ActionListener 接口，主要含有记牌器、电脑点数、玩家点数、存放纸牌的标签控件数组、存储电脑纸牌的向量类对象、CardManager 类对象等属性。主要方法如下：

（1）GrameFrame()：窗体中控件进行初始化。

（2）void actionPerformed(ActionEvent e)：主要处理命令按钮动作，即洗牌、开始游戏、玩家抓牌和现实结果的动作。

4.3 代码实现

Card.java 文件源码：

```
public class Card {
    //代表纸牌面值
    private int value=0;
    //代表纸牌花色
    private int type=0;
    //构造方法，给面值和花色赋值
    public Card(int type,int value){
        this.value=value;
        this.type=type;
```

```
    }
    public int getType() {
        return type;
    }

    public int getValue() {
        return value;
    }

    public void setType(int type) {
        this.type = type;
    }

    public void setValue(int value) {
        this.value = value;
    }
}
```

CardManager.java 文件源码：

```
import javax.swing.JLabel;
import javax.swing.ImageIcon;
import java.awt.Rectangle;
import java.awt.Container;
import java.util.Vector;

public class CardManager {
    public Card[] cards = new Card[52];
   //初始化牌
    public void initCards(){
        for(int i=1;i<=4;i++){
            for (int j = 1; j <= 13; j++){

                cards[(i -1) * 13 + j -1] = new Card(i,j);
            }

        }
    }
     //随机生成牌号
     public void randomCards() {
         Card temp = null;
         //随机生成牌号

         for (int i = 0; i < 52; i++) {
             int a = (int) (Math.random() * 52);
             int b = (int) (Math.random() * 52);
             temp = cards[a];
```

```
                cards[a] = cards[b];
                cards[b] = temp;
            }
        }
        //发牌
    public void gameStart(JLabel game[],Container c){
        //在容器中删除标签组件
        if (game[0] != null) {
                for (int i = 0; i < 52; i++) {
                    c.remove(game[i]);
                }
                c.repaint();
            }
        //在容器中防止 52 个标签组件用于盛放图片
            for (int i = 0; i < 52; i++) {
                game[i] = new JLabel();
                game[i].setBorder(javax.swing.BorderFactory
                                .createEtchedBorder(javax.swing.border.EtchedBorder.
                                                        RAISED));
                game[i].setBounds(new Rectangle(100 + i * 10, 59, 71, 96));
                c.add(game[i]);
            }
            //设置标签组件的图片为 rear.gif，即牌的背面
            for (int i = 0; i < 52; i++) {
                game[i].setIcon(new ImageIcon("images/rear.gif"));
            }
    }
}
```

GameFrame.java 文件源码（对应如图 4-1 所示游戏开始界面）：

```
import javax.swing.JFrame;
import java.awt.Dimension;
import java.awt.Toolkit;
import javax.swing.JLabel;
import java.awt.Rectangle;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.ImageIcon;
import java.util.Vector;
import javax.swing.JOptionPane;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
public class GrameFrame extends JFrame implements ActionListener {
    //洗牌按钮
```

```
JButton clear_btn = new JButton();
//游戏开始按钮
JButton compute_btn = new JButton();
//玩家按钮
JButton game_btn = new JButton();
//游戏结束按钮
JButton gameover_btn = new JButton();
//用于放置 52 张纸牌图片的标签框
JLabel game[] = new JLabel[52];
//定义纸牌管理对象
CardManager cm = new CardManager();
//记录抓牌数
int i = 0;
//定义电脑点数
int computer_dot = 0;
//定义玩家点数
int game_dot = 0;
//存储电脑抓的纸牌
Vector v = new Vector();
JLabel jLabel1 = new JLabel();
JLabel jLabel2 = new JLabel();
  public GrameFrame(){
        getContentPane().setLayout(null);
        this.setTitle("二十一点游戏");
        this.setSize(800, 500);
        //获得屏幕的宽和高
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        //获得当前窗体的宽和高
        Dimension frameSize = this.getSize();
        //设置窗体居中
        if (frameSize.height > screenSize.height) {
              frameSize.height = screenSize.height;
        }
        if (frameSize.width > screenSize.width) {
              frameSize.width = screenSize.width;
        }
        this.setLocation((screenSize.width - frameSize.width) / 2,
                              (screenSize.height - frameSize.height) / 2);
        clear_btn.setBounds(new Rectangle(78, 388, 73, 31));
        clear_btn.setText("洗牌");
        clear_btn.addActionListener(this);
        compute_btn.setBounds(new Rectangle(233, 388, 86, 31));
        compute_btn.setEnabled(false);
        compute_btn.setText("开始游戏");
        compute_btn.addActionListener(this);
        game_btn.setBounds(new Rectangle(413, 389, 91, 32));
        game_btn.setEnabled(false);
```

```
        game_btn.setText("玩家抓牌");
        game_btn.addActionListener(this);
        gameover_btn.setBounds(new Rectangle(625, 390, 91, 32));
        gameover_btn.setEnabled(false);
        gameover_btn.setText("本轮结果");
        gameover_btn.addActionListener(this);
        //定义菜单条
        JMenuBar mb = new JMenuBar();
        //定义菜单
        JMenu mnuFile = new JMenu("文件");
        JMenu mnuHelp = new JMenu("帮助");
        //定义菜单项
        JMenuItem mnuFileExit = new JMenuItem("退出");
        JMenuItem mnuHelpAbout = new JMenuItem("关于…");
        //把菜单条添加到 Frame 窗体上
        this.setJMenuBar(mb);
        jLabel1.setText("电脑显示牌区");
        jLabel1.setBounds(new Rectangle(104, 330, 95, 38));
        jLabel2.setText("用户显示牌区");
        jLabel2.setBounds(new Rectangle(499, 343, 92, 33));
        //把菜单添加到菜单条中
        mb.add(mnuFile);
        mb.add(mnuHelp);
        //把菜单项添加到菜单中
        mnuFile.add(mnuFileExit);
        mnuHelp.add(mnuHelpAbout);
        //对菜单项产生的事件进行注册
        mnuFileExit.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        }
        );
        mnuHelpAbout.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                new AboutFrame();
            }
        }
        );
        this.getContentPane().add(jLabel2);
        this.getContentPane().add(jLabel1);
        this.getContentPane().add(game_btn);
        this.getContentPane().add(clear_btn);
        this.getContentPane().add(gameover_btn);
        this.getContentPane().add(compute_btn);
        this.setVisible(true);
    }
```

```
public static void main(String[] args) {
    GrameFrame    grameframe = new GrameFrame();
}
public void actionPerformed(ActionEvent e) {
    //洗牌按钮
    if (e.getSource() == clear_btn) {
        //关闭和开启相应按钮
        compute_btn.setEnabled(true);
        clear_btn.setEnabled(false);
        //对记牌器、电脑点数和玩家点数进行初始化
        i = 0;
        computer_dot = 0;
        game_dot = 0;
        //把标签框控件数组放入窗体的窗格中
        cm.gameStart(game, this.getContentPane());
        //初始化一副纸牌
        cm.initCards();
        //随机打乱这副纸牌
        cm.randomCards();
    }
    //开始游戏按钮
    if (e.getSource() == compute_btn) {
        //关闭和开启相应按钮
        compute_btn.setEnabled(false);
        game_btn.setEnabled(true);
        //电脑抓牌
        for (int k = 0; k < 20; k++) {
            game[i].setIcon(new ImageIcon("images/rear.gif"));
            game[i].setBounds(new Rectangle(50 + i * 20, 200, 71, 96));
            getContentPane().setComponentZOrder(game[i], 1);
            if (cm.cards[i].getValue() > 10) {
                computer_dot = computer_dot + 1;
            } else {
                computer_dot = computer_dot + cm.cards[i].getValue();
            }
            v.add(cm.cards[i]);
            getContentPane().repaint();
            i = i + 1;
            //如果面值总数大于 15  停止抓牌
            if (computer_dot >= 15) {
                return;
            }
        }
    }
    //玩家抓牌按钮
    if (e.getSource() == game_btn) {
```

```
            //提示玩家
            if (game_dot >= 10) {
                int a = JOptionPane.showConfirmDialog(null,
                        "现在点数为：" + game_dot +
                        "  是否在抓牌",
                        "提示",
                        JOptionPane.YES_NO_OPTION);
                if (a == JOptionPane.NO_OPTION) {
                    game_btn.setEnabled(false);
                    gameover_btn.setEnabled(true);
                    return;
                }
            }
            //设置标签框显示抓到的纸牌
            game[i].setIcon(new ImageIcon("images/" + cm.cards[i].getType() +
                                                  "-" +
                                                  cm.cards[i].getValue() + ".gif"));
            game[i].setBounds(new Rectangle(350 + i * 20, 200, 71, 96));
            this.getContentPane().setComponentZOrder(game[i], 1);
            //计算抓到纸牌的面值
            if (cm.cards[i].getValue() > 10) {
                game_dot = game_dot + 1;
            } else {
                game_dot = game_dot + cm.cards[i].getValue();
            }
            //记录抓到的牌数
            i = i + 1;
            //面值大于 21 停止抓牌，关闭和开启相应按钮
            if (game_dot > 21) {
                game_btn.setEnabled(false);
                gameover_btn.setEnabled(true);
                return;

            }
        }
        //本轮游戏结束按钮
        if (e.getSource() == gameover_btn) {
            //把电脑的纸牌翻过来
            for (int i = 0; i < v.size(); i++) {
                Card card = (Card) v.get(i);
                game[i].setIcon(new ImageIcon("images/" + card.getType() + "-" +
                                                      card.getValue() + ".gif"));
                game[i].setBounds(new Rectangle(50 + i * 20, 200, 71, 96));
                this.getContentPane().setComponentZOrder(game[i], 1);

            }
            //计算胜负
```

```
                String game_over = "";
                if (game_dot > 21 && computer_dot <= 21) {
                    game_over = "电脑获胜";
                } else if (game_dot <= 21 && computer_dot > 21) {
                    game_over = "玩家获胜";
                } else if (game_dot >= 21 & computer_dot >= 21) {
                    game_over = "平局";
                } else if (game_dot > computer_dot) {
                    game_over = "玩家获胜";
                } else if (game_dot < computer_dot) {
                    game_over = "电脑获胜";
                } else if (game_dot == computer_dot) {
                    game_over = "平局";
                }
                //以对话框的方式显示胜负
                String message = "游戏结果\n";
                message = message + "电脑点数：" + String.valueOf(computer_dot) + "\n";
                message = message + "玩家点数：" + String.valueOf(game_dot) + "\n";
                message = message + "游戏结果：" + game_over;
                JOptionPane.showMessageDialog(null, message, "本轮游戏结果",
JOptionPane.INFORMATION_MESSAGE);
                //设置命令按钮可操作
                clear_btn.setEnabled(true);
                compute_btn.setEnabled(true);
                game_btn.setEnabled(true);
                gameover_btn.setEnabled(true);
            }
        }
}
```

AboutFrame.java 源码（对应如图 4-6 所示关于窗体）：

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.Rectangle;
import javax.swing.JTextArea;
import java.awt.Toolkit;
import java.awt.Dimension;

public class AboutFrame extends JFrame {
    JLabel jLabel1 = new JLabel();
    JTextArea ta1 = new JTextArea();
    JLabel jLabel2 = new JLabel();
    JTextArea ta2 = new JTextArea();

    public AboutFrame() {
        getContentPane().setLayout(null);
```

```
            jLabel1.setText("游戏规则");
            jLabel1.setBounds(new Rectangle(27, 0, 97, 36));
            jLabel2.setText("声明");
            jLabel2.setBounds(new Rectangle(26, 176, 80, 27));
            ta2.setEditable(false);
            ta2.setLineWrap(true);
            ta2.setBounds(new Rectangle(26, 207, 398, 63));
            ta1.setColumns(40);
            ta1.setLineWrap(true);
            this.setTitle("关于");
            this.getContentPane().add(jLabel1);
            this.getContentPane().add(ta1);
            this.getContentPane().add(jLabel2);
            this.getContentPane().add(ta2);
            ta1.setEditable(false);
            ta1.setText("电脑先抓牌，玩家后抓牌。计算自己的面值总数，比较面值数，如果面值
总数都大于或等于 21，则平局；如果玩家和电脑的面值总数有一个大于 21 点，另一个不大于 21 点，
则不大于 21 点的，为赢家；如果玩家和电脑的面值总数都不大于 21，则面值总数大的为赢家。");
            ta1.setBounds(new Rectangle(25, 36, 392, 130));
            ta2.setText("该游戏中，纸牌的图片来自于 Windows XP 的纸牌游戏，图片权属于原作者
所有！ ");
            this.setSize(450, 300);
            Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
            //获得当前窗体的宽和高
            Dimension frameSize = this.getSize();
            //设置窗体居中
            if (frameSize.height > screenSize.height) {
                frameSize.height = screenSize.height;
            }
            if (frameSize.width > screenSize.width) {
                frameSize.width = screenSize.width;
            }
            this.setLocation((screenSize.width - frameSize.width) / 2,
                             (screenSize.height - frameSize.height) / 2);
            this.setVisible(true);

        }
    }
```

4.4 运行与发布

4.4.1 运行

将 Card.java、CardManager.java、GameFrame.java 和 AboutFrame.java 4 个文件保存到一个文件夹中，如 e:\java\game。在使用 javac 命令进行编译之前，应使用如下命令设

置类路径：

```
e:\java\game>set classpath= e:\java\game
```

然后利用 javac 命令对文件进行编译，使用如下命令：

```
Javac GameFrame.java
```

之后，使用 java 执行程序：

```
Java GameFrame
```

程序即运行。

4.4.2 发布

使用 jar.exe 将应用程序打包，把应用程序中涉及的类和图片压缩成一个 jar 文件，这样就可以发布程序了。

（1）编写一个程序文件，名称为 MANIFEST.MF，代码如下：

```
Manifest –Version: 1.0
Created –By: 1.5.0_02 (Sun Microsystems Inc.)
Main-Class: GameFrame
MANIFEST.MF 文件保存在 e:\java\game
```

（2）使用如下命令生成 jar 文件：

```
jar cfm GameFrame.jar MANIFEST.MF *.class
```

（3）编写一个 bat 文件，文件名为 GameFrame.bat。

内容为：javaw –jar GameFrame.jar

与 GameFrame.jar 保存在同一个文件夹下。

（4）运行 GameFrame.jar 文件即可。前提是计算机上安装了 Java JDK，并配置了环境变量。

4.5 本项目实现中常见问题

（1）定义控件数组后，在使用控件数组时，应创建控件对象赋给控件数组元素对象，即：

//定义控件数组

```
JLabel    jlabel[]=new JLabel[10];
```

//使用控件数组元素，必须创建

```
jlabel[0]=new JLabel();
```

（2）Vector 向量中提取的对象要进行恢复原型，进行强制类型转换。

（3）菜单项应放入菜单中，菜单应放入菜单条中，菜单条应放入窗体中，顺序应正确，否则会与设计的菜单不一样。

（4）设置 Frame 对象的可见性，this.setVisible(true)；语句放置在构造方法的最后一条语句。

（5）发布前，images 图片文件夹，存于工程目录下；发布后，images 图片文件夹和 GameFrame.jar 文件在同一目录下。

4.6 项目技术支持

4.6.1 菜单

制作菜单分成三部分：菜单条、菜单和菜单项。菜单条相当于一个容器用来承装菜单，而菜单项是放入到菜单中的，菜单条放入容器中，就能看到 Windows 程序的普通菜单。Java 中制作菜单用到的类为 JMenuBar、JMenu、JMenuItem。在 java.swing 包。JMenuBar 类代表菜单条，JMenu 类代表菜单，JmenuItem 类代表菜单项。

1．JMenuBar类常用方法

JMenuBar() 创建新的菜单条。
JMenu add(JMenu c)将指定的菜单追加到菜单条的末尾。
JMenu getMenu(int index)返回菜单条中指定位置的菜单。
int getMenuCount()返回菜单条上的菜单数。

2．JMenu类常用方法

JMenu()创建没有标题的新菜单。
Jmenu(String s)创建带有标题的一个新菜单。
Jmenu(String s, boolean b)创建一个含有标题的新菜单，指定其是否为分离式菜单。
JMenuItem add(JMenuItem menuItem)将某个菜单项追加到此菜单的末尾。
JMenuItem add(String s)创建具有指定文本的菜单项，并将其追加到此菜单的末尾。
void addMenuListener(MenuListener l)添加菜单事件的侦听器。
void addSeparator()将新分隔符追加到菜单的末尾。
void remove(JMenuItem item)从此菜单移除指定的菜单项。

3．JMenuItem类常用方法

JMenuItem()创建不带有设置文本或图标的菜单项。

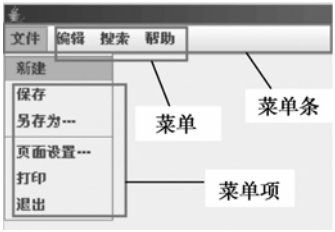

图 4-7 菜单制作

JmenuItem(Icon icon)创建带有指定图标的菜单项。
void addMenuKeyListener(MenuKeyListener l) 将 MenuKeyListener 添加到菜单项。
void setEnabled(boolean b)启用或禁用菜单项。

4．实例

制作如图 4-7 所示菜单。关键代码如下：

```
//定义菜单条
JMenuBar mb = new JMenuBar();
//定义菜单
JMenu mnuFile = new JMenu("文件");
JMenu mnuEdit = new JMenu("编辑");
JMenu mnuSearch = new JMenu("搜索");
JMenu mnuHelp = new JMenu("帮助");
```

```
//定义菜单项
JMenuItem mnuFileNew = new JMenuItem("新建");
JMenuItem mnuFileOpen = new JMenuItem("打开…");
JMenuItem mnuFileSave = new JMenuItem("保存");
JMenuItem mnuFileSaveAs = new JMenuItem("另存为…");
JMenuItem mnuFilePageSetting = new JMenuItem("页面设置…");
JMenuItem mnuFilePrint = new JMenuItem("打印");
JMenuItem mnuFileQuit = new JMenuItem("退出");

//把菜单条添加到 Frame 窗体上
this.setJMenuBar(mb);
//向菜单添加菜单项
mnuFile.add(mnuFileNew);
mnuFile.add(mnuFileSave);
mnuFile.add(mnuFileSaveAs);
mnuFile.addSeparator(); //添加分割条
mnuFile.add(mnuFilePageSetting);
mnuFile.add(mnuFilePrint);
mnuFile.add(mnuFileQuit);

//把菜单添加到菜单条中
mb.add(mnuFile);
mb.add(mnuEdit);
mb.add(mnuSearch);
mb.add(mnuHelp);
```

4.6.2 Vector向量类

Vector 在 java.util 包中，可以实现可增长的对象数组。与数组一样，它包含可以使用整数索引进行访问。Vector 的大小可以根据实际的需要增大或缩小，以适应创建 Vector 后进行添加或移除项的操作。

1．构造方法

Vector() 构造一个空向量，使其内部数据数组的大小为 10，其标准容量增量为零。

Vector(Collection<? extends E> c) 构造一个包含指定集合中的元素的向量，这些元素按其集合的迭代器返回元素的顺序排列。

Vector(int initialCapacity) 使用指定的初始容量和等于零的容量增量构造一个空向量。

Vector(int initialCapacity, int capacityIncrement) 使用指定的初始容量和容量增量构造一个空的向量。

2．常用方法

boolean add(Object o)：将指定元素追加到此向量的末尾。

void clear()：从此向量中移除所有元素。

Object get(int index)：返回向量中指定位置的元素。

boolean isEmpty()：测试此向量是否不包含组件。
Object remove(int index)：移除此向量中指定位置的元素。
int size()：返回此向量中的组件数。

3．实例

把三个学生信息放入 Vector 向量类中，并重 Vector 中遍历。

```
class Student{
      private String no;
      private String name;
      public void setNo(String no){
            this.no=no;
      }
      public String getNo(){
           return this.no;
      }
      public void setName(String name)
            this.name=name;
      }
      public String getName(){
           return this.name;
      }
}
import java.util.Vector
public class DemoVector{
   public static void main(String[] args){
         Vector v=new Vector();
         Student s1=new Student("张海峰","070111");
         Student s2=new Student("李晶","070112");
         Student s3=new Student("王海","070113");
         //把 Student 对象添加到向量类 V 对象中
         v.add(s1);
         v.add(s2);
         v.add(s3);
         //遍历向量类中存储的数据
      for(int i=0;i<v.size();i++){
          //把向量中的 Object 类型数据强制类型转换
          Student s=(Student)v.get(i);
          System.out.println("姓名:"+s.getName()+"    学号"+s.getNo());
          }
   }
}
```

4.6.3 集合类简介

集合类存放于 java.util 包中。集合类存放的都是对象的引用，而非对象本身，出于表达

上的便利，集合中的对象就是指集合中对象的引用（reference）。

java.util 中共有 13 个类可用于管理集合对象，它们支持集、列表或映射等集合。集合类型主要有 3 种：Set（集）、List（列表）和 Map（映射）。

Java 中的集合类都实现了 Collection 接口，其中类继承结构如下：

```
Collection<--List<--Vector
Collection<--List<--ArrayList
Collection<--List<--LinkedList
Collection<--Set<--HashSet
Collection<--Set<--HashSet<--LinkedHashSet
Collection<--Set<--SortedSet<--TreeSet
```

其中 List 和 Set 都是接口。

1．集（Set）

集（Set）是最简单的一种集合，它的对象不按特定方式排序，只是简单地把对象加入集合中，就像往口袋里放东西。对集中成员的访问和操作是通过集中对象的引用进行的，所以集中不能有重复对象。集也有多种变体，可以实现排序等功能，如 TreeSet，它把对象添加到集中的操作变为按照某种比较规则将其插入到有序的对象序列中。它实现的是 SortedSet 接口，也就是加入了对象比较的方法。通过对集中的对象迭代，可以得到一个升序的对象集合。

HashSet：使用 HashMap 的一个集的实现。虽然集定义成无序，但必须存在某种方法能相当高效地找到一个对象。使用一个 HashMap 对象实现集的存储和检索操作是在固定时间内实现的。

TreeSet：在集中以升序对对象排序的集的实现。这意味着从一个 TreeSet 对象获得第一个迭代器将按升序提供对象。TreeSet 类使用了一个 TreeMap。

2．列表（list）

列表的主要特征是其对象以线性方式存储，没有特定顺序，只有一个开头和一个结尾，当然，它与根本没有顺序的集是不同的。列表在数据结构中分别表现为：数组和向量、链表、堆栈、队列。关于实现列表的集合类，是日常工作中经常用到的。

Vector：实现一个类似数组一样的表，自动增加容量来容纳你所需的元素。使用下标存储和检索对象就像在一个标准的数组中一样。也可以用一个迭代器从一个 Vector 中检索对象。Vector 是唯一的同步容器类，当两个或多个线程同时访问时也是性能良好的。

Stack：这个类从 Vector 派生而来，并且增加了方法实现栈，是一种后进先出的存储结构。

LinkedList：实现一个链表。由这个类定义的链表也可以像栈或队列一样被使用。

ArrayList：实现一个数组，它的规模可变并且能像链表一样被访问。它提供的功能类似 Vector 类但不同步。

3．映射（Map）

映射与集或列表有明显区别，映射中每个项都是成对的。映射中存储的每个对象都有一个相关的关键字（Key）对象，关键字决定了对象在映射中的存储位置，检索对象时必须提供相应的关键字，就像在字典中查单词一样。关键字应该是唯一的。

关键字本身并不能决定对象的存储位置，它需要通过一种散列（hashing）技术来处理，

产生一个被称做散列码（hashcode）的整数值，散列码通常用做一个偏置量，该偏置量是相对于分配给映射的内存区域起始位置的，由此确定关键字/对象对的存储位置。理想情况下，散列处理应该产生给定范围内均匀分布的值，而且每个关键字应得到不同的散列码。

HashTable：实现一个映射，所有的键必须非空。为了能高效地工作，定义键的类必须实现 hashcode()方法和 equal()方法。这个类是前面 Java 实现的一个继承，并且通常能在实现映射的其他类中更好地使用。

HashMap：实现一个映射，允许存储空对象，而且允许键是空（由于键必须是唯一的，只能有一个）。

WeakHashMap：实现如果一个键对一个对象而言不再被引用，键/对象对将被舍弃的映射。这与 HashMap 形成对照，映射中的键维持键/对象对的生命周期，尽管使用映射的程序不再有对键的引用，并且因此不能检索对象。

TreeMap：实现对象是按照键升序排列的这样一个映射。

4.6.4 ImageIcon组件

ImageIcon 类在 javax.swing 包中，实现了 Icon 接口，根据 Image 来绘制 Icon。

1．构造方法

ImageIcon()：创建一个未初始化的图像图标对象。
ImageIcon(Image image)：根据图像对象创建一个图像图标对象。
ImageIcon(Image image, String description)：根据图像创建一个图像图标对象。
ImageIcon(String filename)：根据指定的文件创建一个图像图标对象。
ImageIcon(String filename, String description)：根据指定的文件创建一个图像图标对象。

2．常用方法

String getDescription()：获得图像的描述。
int getIconHeight()：获得图标的高度。
int getIconWidth()：获得图标的宽度。
Image getImage()：返回此图标的 Image。
protected void loadImage(Image image)：加载图像，并且只在图像已加载时返回。
void setDescription(String description)：设置图像的描述。
void setImage(Image image)：设置由此图标显示的图像。

3．实例

通过文件选择器，选择图片，显示在窗体上。

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.Rectangle;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JFileChooser;
```

```
import javax.swing.filechooser.FileFilter;
import javax.swing.ImageIcon;
import java.io.File;

public class ImageFrame extends JFrame implements ActionListener {
    JLabel lbl = new JLabel();
    JButton btn_open = new JButton();

    public ImageFrame() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }

    private void jbInit() throws Exception {
        getContentPane().setLayout(null);
        lbl.setBounds(new Rectangle(68, 31, 375, 216));
        this.setTitle("打开图片");
        btn_open.addActionListener(this);
        this.getContentPane().add(lbl);
        this.getContentPane().add(btn_open);
        but_open.setBounds(new Rectangle(189, 327, 145, 43));
        but_open.setText("打开图片");
        this.setSize(500, 500);
        this.setVisible(true);
    }

    public static void main(String[] args) {
        ImageFrame imageframe = new ImageFrame();
    }

    public void actionPerformed(ActionEvent e) {
        //创建文件选择器，默认打开当前路径
        JFileChooser file = new JFileChooser(".");
        int result = file.showOpenDialog(null);
        /*使用 showOpenDialog()方法，显示出打开选择文件的窗口，
          选择了某个文件，返回值为 0。
          返回值为 1 代表选择了取消按钮或者直接关闭了窗口。*/
        if (result == 0) {
            //获得你选择的文件绝对路径
            String path = file.getSelectedFile().getAbsolutePath();
            lbl.setIcon(new ImageIcon(path));
        }
    }
}
```

4.6.5 Toolkit组件

Toolkit 类在 java.awt 包下，是 AWT（Abstract Window Toolkit）的所有实际实现的抽象父类。Toolkit 用于把各种组件绑定到特定的本地工具箱实现上。大多数应用不应直接调用该类中的任何方法。Toolkit 定义的方法是“胶合剂”，将 java.awt 包中独立于平台的类与 java.awt.peer 中的对应物连接起来。Toolkit 定义的一些方法能直接查询本地操作系统。

1．构造方法

Toolkit()

2．常用方法

abstract void beep() 发出一个音频嘟嘟声。

static Toolkit getDefaultToolkit() 获取默认工具包。

abstract Clipboard getSystemClipboard() 获取系统 Clipboard 的一个实例，该 Clipboard 与本机平台提供的剪贴板工具相互作用。

图 4-8 实现蜂鸣窗体

3．实例

制作如图 4-8 所示窗体，现实窗体居中，单击窗体上的按钮发出蜂鸣声。

```
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.Toolkit;
import java.awt.Dimension;

public class Frame2 extends JFrame implements ActionListener {
    public Frame2() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        getContentPane().setLayout(null);
        this.setTitle("蜂鸣声");
        btn_set.setBounds(new Rectangle(100, 113, 160, 50));
        btn_set.setToolTipText("");
        btn_set.setText("蜂鸣声");
        btn_set.addActionListener(this);
        this.getContentPane().add(btn_set);
```

```
        //获得屏幕的宽和高
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        //获得当前窗体的宽和高
        this.setSize(300,300);
        Dimension frameSize = this.getSize();
        //设置窗体居中
        if (frameSize.height > screenSize.height) {
            frameSize.height = screenSize.height;
        }
        if (frameSize.width > screenSize.width) {
            frameSize.width = screenSize.width;
        }
        this.setLocation((screenSize.width - frameSize.width) / 2,
                         (screenSize.height - frameSize.height) / 2);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        Frame2 frame2 = new Frame2();
    }
    JButton btn_set = new JButton();
    public void actionPerformed(ActionEvent e) {
        Toolkit.getDefaultToolkit().beep();
    }
}
```

4.7 实训

1. 题目

二十一点游戏。

2. 要求

（1）发牌变成电脑发一张、玩家发一张。
（2）电脑第一张牌显示面值，其余牌不显示面值。
（3）J、Q、K 牌的面值按照 10 计算。
（4）实现积分统计，平局计 1 分，赢计 3 分，输计 0 分。

第 5 章　学生信息管理系统

5.1　项目目标

该项目开发的系统为学生信息管理系统软件，是鉴于目前学校快速发展，学校规模越来越大，学校与时俱进，课程不断改革，学生数量与课程数量都在迅速的增长，学生信息呈爆炸性增长的前提下，原始的手工管理耗费学生与工作人员大量的时间和精力，而效率与准确性却很低，学校对学生信息管理的自动化与准确化的要求日益强烈，为满足学校管理学生信息的需要，设计并完成该系统。系统完成后可用于学校学生信息管理，能够实现对学生信息进行存储、查询、修改等功能。

5.2　项目需求分析

根据学校管理学生信息的需求，分析学生信息管理系统的功能，列出主要功能如表 5-1 所示。

表 5-1　功能列表

角色名	功　能　名	需要实现的功能
用户	学生信息管理	1．可以添加学生信息 2．可以删除学生信息 3．可以修改学生信息
	选课	可以通过输入学号，选择课程，进行选课
	课程信息管理	1．可以添加课程信息 2．可以删除课程信息 3．可以修改课程信息
	成绩管理	1．根据学生的选课，进行成绩的录入 2．修改学生所选课程的成绩
	查询	1．可以通过输入学生学号、姓名、性别、学院、专业查询学生信息 2．可以通过输入课程名、授课教师，查询课程信息 3．可以根据学号，查询所选课程的成绩

5.3 概要设计

5.3.1 架构设计

为方便开发与维护，本系统采用两层架构：视图层和业务逻辑层。视图层用来接收数据、显示结果、数据验证和调用业务逻辑等；视图层是用户看到并与之交互的界面，与用户直接接触。业务逻辑层主要进行业务逻辑处理，与数据库接触。架构模型如图 5-1 所示。

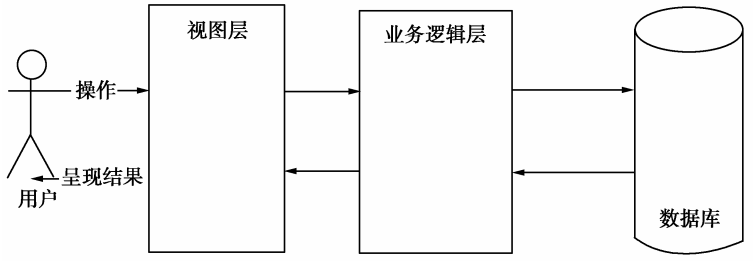

图 5-1　架构模型

为使系统更便于开发与维护，业务逻辑层由两部分组成：实体类与管理类。实体类与数据库中的字段一一对应，方便对数据库的操作；操作逻辑由管理类实现。应用技术模式设计如图 5-2 所示。

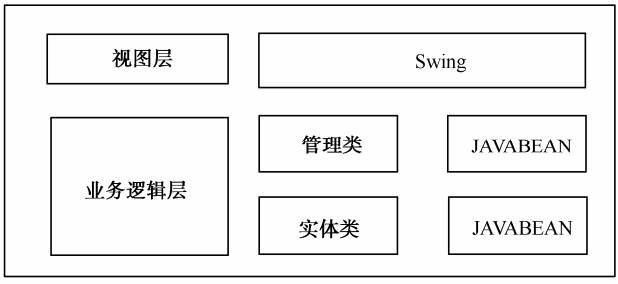

图 5-2　应用技术模式设计

5.3.2 功能分配

本系统将功能列表中的功能整合、划分为三个主要功能模块：学生管理模块、课程管理模块、成绩管理模块。

模块功能划分如图 5-3 所示。

5.3.3 功能、业务流程设计

1．添加学生信息

用户输入学生信息（包括姓名、性别、民族、籍贯、出生日期、入学年份、专业、学院等），用户核查无误后确认，将数据添加到数据库中。

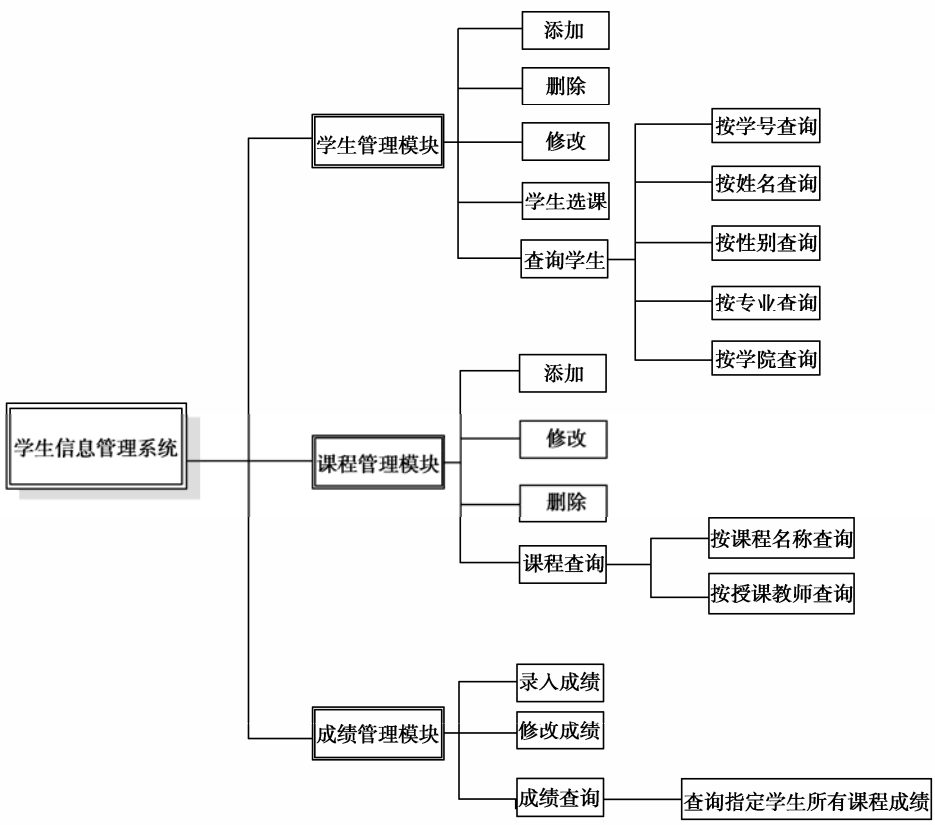

图 5-3　模块功能划分

2．修改学生信息

输入学生学号，查询并显示学生信息，除学号外，其他信息可以修改。修改学生信息，用户核查无误后确认，并将修改后的数据保存到数据库中。

3．删除学生信息

输入学生学号，查询并在数据库中删除该记录。

4．学生选课

在提示列表中，选择学生学号和该名学生要选择的课程，显示所选课程详细信息（包括授课教师、上课时间、上课地点和课程类别等），用户核查无误后确认，将新的选课信息保存到数据库中。

5．查询学生信息

可以通过输入学生学号、姓名、性别、所属学院进行查询，显示出所有符合条件的学生信息。

6．添加课程信息

用户输入课程信息（包括课程名称、授课教师、上课时间、上课地点和课程类别等），用户核查无误后确认，将新的课程信息保存到数据库中。

7．修改课程信息

输入课程号，查询并显示课程信息。修改课程信息，用户核查无误后确认，并将修改后的数据保存到数据库中。

8．删除课程信息

输入课程编号，查询并在数据库中删除该记录。

9．课程查询

可以通过输入课程名称、授课教师姓名查询课程详细信息，显示出所有符合条件的课程信息。

10．录入成绩

输入学生学号及所选课程，显示课程详细信息，只有成绩可以输入。录入成绩，确认并保存到数据库中。

11．修改成绩

输入学生学号及所选课程，显示课程详细信息，只有成绩可以修改，修改成绩后，确认并更新数据库中成绩信息数据。

12．查询成绩

输入学生的学号查询该学生所有课程的成绩。

5.3.4 对象模型

分析数据需求，获得业务对象，如图 5-4 所示。

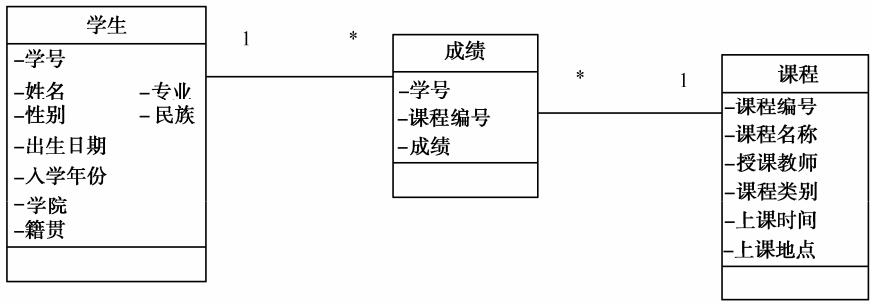

图 5-4 业务对象

5.4 详细设计与代码实现

5.4.1 数据库设计与实现

1．表的设计

根据对业务对象模型的分析，设计数据库中包含三个表：学生信息表（student）、课程信息表（course）和学生选课及成绩表（grade）。

编制数据字典如表 5-2、表 5-3 和表 5-4 所示。

表 5-2 学生（student）信息

汉语名称	英文名称	类型	长度	默认值	PK/FK	是否允许为空
学号	sno	varchar	10		PK	否
学生姓名	sname	varchar	10			否
性别	ssex	varchar	3	男		
民族	sethnix	varchar	6	汉		
籍贯	shome	varchar	10			
入学年份	syear	varchar	10			
专业	smajor	varchar	10			
学院	scollege	varchar	10			
出生日期	sbirth	datetime				

表 5-3 课程（course）信息

汉语名称	英文名称	类型	长度	默认值	PK/FK	是否允许为空
课程编号	cno	varchar	10		PK	否
课程名称	cname	varchar	12			否
授课教师	cteacher	varchar	12			
课程类别	ctype	varchar	10			
上课地点	cplace	varchar	12			
上课时间	ctime	datetime				

表 5-4 学生选课及成绩（grade）信息

汉语名称	英文名称	类型	长度	默认值	PK/FK	是否允许为空
学号	sno	varchar	10		PK	否

课程编号	cno	varchar	10		PK	否
成绩	grade	int				

2．创建数据表

本系统连接的数据库为 Oracle 数据库，打开 Oracle，创建表如下。

（1）创建学生（student）信息，代码如下：

```
create table student(
sno varchar(10) primary key,          //学号，主键
sname varchar(10),                    //学生姓名
ssex varchar(3),                      //性别
sethnix varchar(6),                   //民族
shome varchar(50),                    //籍贯
syear varchar(10),                    //入学年份
smajor varchar(10),                   //专业
scollege varchar(10),                 //学院
sbirth datetime                       //出生日期
);
```

（2）创建课程（course）信息，代码如下：

```
create table course(
cno varchar(10) primary key,          //课程编号，主键
cname varchar(12),                    //课程名称
cteacher varchar(12),                 //授课教师
ctype varchar(10),                    //课程类别
cplace varchar(12),                   //上课地点
ctime datetime                        //上课时间
);
```

（3）创建学生选课及成绩（grade）信息，代码如下：

```
create table grade(
sno varchar(10),                      //学号
cno varchar(10),                      //课程编号
grade int,                            //成绩
constraint pk_grade primary key(sno,cno)   //学号与课程编号为联合主键
)
```

5.4.2 包的设计与类的管理

本系统采用两层架构，视图层和业务逻辑层，其中业务逻辑层分为两部分——实体类和管理类，为方便管理，将其分别放置在不同的包中，其中实体类放在 cvit.com.model 包中，管理类放在 cvit.com. manager 包中，视图层放置在 cvit.com.view 包中，本系统还需要一个用于连接数据库的公共类，放置在 cvit.com.pub 包中。

工程结构如图 5-5 所示。

下面一一说明各部分的设计与实现。

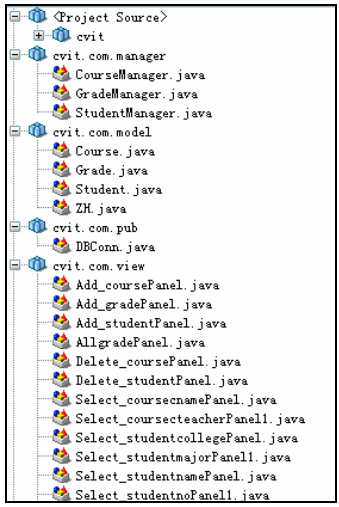

图 5-5　工程结构

5.4.3　业务逻辑层之实体类的设计与实现

实体类是与数据库关系最密切的层，其中包含的类与数据库中的表一一对应，类的属性与数据库中字段一一对应，而方法则主要是 get 和 set 方法。

其主要功能是为实现对数据库表的增、删、改、查等操作，提供相应的对象。

根据数据字典已知，数据库中将建立三个表，则实体类包含与之一一对应的三个类，分别是：Student 类，与学生（student）信息表对应；Course 类，与课程（course）信息表对应；Grade 类，与学生选课及成绩（grade）信息表对应。另外，根据业务对象模型可知，数据库的三个表之间存在关联，在对表做增、删、改、查等操作时，存在同时对多表进行操作，因此还需要一个综合类，用于封装多表的信息。

根据以上分析，实体类包含三个类，分别放在三个 java 文件中，还需要一个综合类，为方便管理，所有实体类及综合类都放置在 cvit.com.model 包下。

1．Student.java

Student 类与学生（student）信息表对应，其属性与学生（student）信息表中的字段一一对应，包含 9 个属性及其 get 和 set 方法。

其代码如下：

```
package cvit.com.model;//包
//学生类
public class Student {
        public String sno;          //学号
        public String sname;        //学生姓名
        public String ssex ;        //学生性别
        public String sethnix ;     //民族
        public String shome ;       //籍贯
        public String syear ;       //入学时间
```

```
        public String smajor;       //专业
        public String scollege ;    //所在分院
        public String sbirth;       //出生日期
    //以下是 set 和 get 方法
    public String getSbirth() {
        return sbirth;
    }
    public String getScollege() {
        return scollege;
    }
    public String getSethnix() {
        return sethnix;
    }
    public String getShome() {
        return shome;
    }
    public String getSmajor() {
        return smajor;
    }
    public String getSname() {
        return sname;
    }
    public String getSno() {
        return sno;
    }
    public String getSsex() {
        return ssex;
    }
    public String getSyear() {
        return syear;
    }
    public void setSbirth(String sbirth) {
        this.sbirth = sbirth;
    }
    public void setScollege(String scollege) {
        this.scollege = scollege;
    }
    public void setSethnix(String sethnix) {
        this.sethnix = sethnix;
    }
    public void setShome(String shome) {
        this.shome = shome;
    }
    public void setSmajor(String smajor) {
        this.smajor = smajor;
    }
    public void setSname(String sname) {
```

```
        this.sname = sname;
    }
    public void setSno(String sno) {
        this.sno = sno;
    }
    public void setSsex(String ssex) {
        this.ssex = ssex;
    }
    public void setSyear(String syear) {
        this.syear = syear;
    }
}
```

2．Course.java

Course 类与课程（course）信息表对应，其属性与课程（course）信息表中的字段一一对应，包含 6 个属性及其 get 和 set 方法。

其代码如下：

```
package cvit.com.model;     //包
//课程类
public class Course {
    public String cno;      //课程编号
    public String cname;    //课程名称
    public String cteacher; //任课教师
    public String ctype;    //科目类型
    public String cplace;   //上课地点
    public String ctime;    //上课时间
//以下是 set 和 get 方法
    public void setCname(String cname) {
        this.cname = cname;
    }
    public void setCno(String cno) {
        this.cno = cno;
    }
    public void setCplace(String cplace) {
        this.cplace = cplace;
    }
    public void setCteacher(String cteacher) {
        this.cteacher = cteacher;
    }
    public void setCtime(String ctime) {
        this.ctime = ctime;
    }
    public void setCtype(String ctype) {
        this.ctype = ctype;
    }
```

```
    public String getCtype() {
        return ctype;
    }
    public String getCtime() {
        return ctime;
    }
    public String getCteacher() {
        return cteacher;
    }
    public String getCplace() {
        return cplace;
    }
    public String getCno() {
        return cno;
    }
    public String getCname() {
        return cname;
    }
}
```

3．Grade.java

Grade 类与学生选课及成绩（grade）信息表对应，其属性与学生选课及成绩（grade）信息表中的字段一一对应，包含 3 个属性及其 get 和 set 方法。

其代码如下：

```
package cvit.com.model;
//成绩类
public class Grade {
    public String sno;      //学生学号
    public String cno;      //课程编号
    public String grade;    //成绩
//以下是 set 和 get 方法
    public String getCno() {
        return cno;
    }
    public String getGrade() {
        return grade;
    }
    public String getSno() {
        return sno;
    }
    public void setCno(String cno) {
        this.cno = cno;
    }
    public void setGrade(String grade) {
        this.grade = grade;
```

```
    }
    public void setSno(String sno) {
        this.sno = sno;
    }
}
```

4．ZH.java

ZH 类用于综合查询，分别对应学生（student）信息表中的学号、姓名字段，课程（course）信息表中的课程编号和课程名称字段，学生选课及成绩（grade）信息表中的成绩字段，包括 5 个属性及其 get 和 set 方法。

其代码如下：

```
package cvit.com.model;
//综合查询类
public class ZH {
    public String sno;      //学生学号
    public String sname;    //学生姓名
    public String cno;      //课程编号
    public String cname;                                    //课程名称
    public String grade;                                    //成绩
//以下是 set 和 get 方法
    public String getCname() {
        return cname;
    }
    public String getCno() {
        return cno;
    }
    public String getGrade() {
        return grade;
    }
    public String getSname() {
        return sname;
    }
    public String getSno() {
        return sno;
    }
    public void setCname(String cname) {
        this.cname = cname;
    }
    public void setCno(String cno) {
        this.cno = cno;
    }
    public void setGrade(String grade) {
        this.grade = grade;
    }
    public void setSname(String sname) {
```

```
        this.sname = sname;
    }
    public void setSno(String sno) {
        this.sno = sno;
    }
}
```

5.4.4 连接数据库公共类设计与实现

实体类用于提供操作数据库的对象，要操作数据库，需要使用 JDBC 连接数据库。本系统设计一个公共类 DBConn，用于实现连接数据库的功能，放置在 cvit.com.pub 包中，文件名为 DBConn.java。

其代码如下：

```
package cvit.com.pub;
import java.sql.*;
public class DBConn {                                   //连接数据库
    private String driver = "oracle.jdbc.driver.OracleDriver";
    private String url = "jdbc:oracle:thin:@localhost:1521:ora";
    private Connection conn = null;                     //连接数据库对象
    private String user = "scott";                      //用户名
    private String pwd = "tiger";                       //口令
    public Connection getConn() {
        try {
            Class.forName(driver);
        } catch (ClassNotFoundException ex) {
            System.out.println("加载驱动程序有错误，驱动程序类不存在");
        }
        try {
            conn = DriverManager.getConnection(url, user, pwd);
        } catch (SQLException ex1) {
            System.out.print("取得连接的时候有错误，请核对用户名和密码");
        }
        return conn;
    }
    //测试是否连接数据库
    public static void main(String args[]) {
        DBConn db = new DBConn();
        Connection conn = db.getConn();
        System.out.println("Ok");
    }
}
```

5.4.5 业务逻辑层之管理类设计与实现

管理类用于接受用户的输入并调用实体类对数据库进行持久化操作，调用视图层去响应用户的操作、满足用户的需求，是整个系统最核心的部分。

为方便管理，将控制层中的所有类，放置在 cvit.com.manager 包中。

针对三个实体类，设计三个管理类，用于管理三个实体类，其中 StudentManager 类管理 Student 类，放在 StudentManager.java 文件中；CourseManager 类管理 Course 类，放在 CourseManager.java 文件中；GradeManager 类管理 Grade 类，放在 GradeManager.java 文件中。

1．StudentManager.java

StudentManager 类用于管理 Student 类，主要包含：添加学生信息的方法——addStudent；删除学生信息的方法——deleteUser；修改学生信息的方法——update_Student；根据学号查询学生信息的方法——findStudent；根据姓名查询学生信息的方法——findStudentname；根据性别查询学生信息的方法——findStudentsex；根据专业查询学生信息的方法——findStudenmajor 等方法。

其代码如下：

```
package cvit.com.manager;
import java.sql.ResultSet;
import cvit.com.pub.DBConn;
import java.sql.*;
import java.sql.Connection.*;
import cvit.com.model.*;
import cvit.com.*;
import java.util.Vector;
//学生信息管理类
public class StudentManager {
    //添加学生信息
    public int addStudent(Student s) {
    Connection conn = null;                                    //连接数据库对象
    Statement stmt = null;                                     //执行 sql 语句的对象
    ResultSet rs = null;                                       //执行 sql 语句的返回结果
        DBConn db = new DBConn();
        conn = db.getConn();
        int i = 0;
        String sql = "insert into student values('" + s.getSno() + "','" +
                    s.getSname() + "','" + s.getSsex() + "','" + s.getSethnix() +
                    "','" + s.getShome() + "','" + s.getSyear() + "','" +
                    s.getSmajor() + "','" + s.getScollege() + "','" +
                    s.getSbirth() + "')";
        try {
            stmt = conn.createStatement();
            i = stmt.executeUpdate(sql);
            stmt.close();
            conn.close();
        } catch (Exception e1) {
            e1.printStackTrace();
        }
        return i;
```

```
}
//删除学生信息
public int deleteUser(String sno) {
Connection conn = null;                                    //连接数据库对象
Statement stmt = null;                                     //执行 sql 语句的对象
ResultSet rs = null;                                       //执行 sql 语句的返回结果
    DBConn db = new DBConn();
    conn = db.getConn();
    int i = 0;
    String sql = "delete from student where sno='" + sno + "'";
    try {
        stmt = conn.createStatement();
        i = stmt.executeUpdate(sql);
        stmt.close();
        conn.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return i;
}
//修改学生信息
public int update_Student(String sno, String sname, String ssex,
                          String sethnix, String shome, String syear,
                          String smajor, String scollege, String sbirth) {
 Connection conn = null;                                   //连接数据库对象
Statement stmt = null;                                     //执行 sql 语句的对象
ResultSet rs = null;                                       //执行 sql 语句的返回结果
    DBConn db = new DBConn();
    conn = db.getConn();
    int i = 0;
    String sql = "update student set sname='" + sname +"',
                 ssex='" + ssex + "',sethnix='" + sethnix +"',
                 shome='" + shome + "',syear='" + syear +"',
                 smajor='" + smajor + "',scollege='" + scollege +"',
                 sbirth='" + sbirth + "' where sno='" +
                 sno + "'";

    try {
        stmt = conn.createStatement();
        i = stmt.executeUpdate(sql);
        stmt.close();
        conn.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return i;
}
```

```
//根据学号查询学生信息
public Student findStudent(String sno) {
Connection conn = null;                                    //连接数据库对象
Statement stmt = null;                                     //执行 sql 语句的对象
ResultSet rs = null;                                       //执行 sql 语句的返回结果
String sql = "select *from student where sno='" + sno + "'";
    Student s = null;
    DBConn db = new DBConn();
    conn = db.getConn();
    try {
        stmt = conn.createStatement();
        rs = stmt.executeQuery(sql);
        if (rs.next()) {
            s = new Student();
            s.setSno(rs.getString("sno"));
            s.setSname(rs.getString("sname"));
            s.setSsex(rs.getString("ssex"));
            s.setSethnix(rs.getString("sethnix"));
            s.setShome(rs.getString("shome"));
            s.setSyear(rs.getString("syear"));
            s.setSmajor(rs.getString("smajor"));
            s.setScollege(rs.getString("scollege"));
            s.setSbirth(rs.getString("sbirth"));
        }
        rs.close();
        stmt.close();
        conn.close();
    }
    catch (SQLException ex3) {
        ex3.printStackTrace();
    }
    return s;
}
//根据姓名查询学生信息
public Student findStudentname(String sname) {
Connection conn = null;                                    //连接数据库对象
Statement stmt = null;                                     //执行 sql 语句的对象
ResultSet rs = null;                                       //执行 sql 语句的返回结果
    String sql = "select *from student where sname='" + sname + "'";
    Student s = null;
    DBConn db = new DBConn();
    conn = db.getConn();
    try {
        stmt = conn.createStatement();
        rs = stmt.executeQuery(sql);
        if (rs.next()) {
            s = new Student();
```

```
                s.setSno(rs.getString("sno"));
                s.setSname(rs.getString("sname"));
                s.setSsex(rs.getString("ssex"));
                s.setSethnix(rs.getString("sethnix"));
                s.setShome(rs.getString("shome"));
                s.setSyear(rs.getString("syear"));
                s.setSmajor(rs.getString("smajor"));
                s.setScollege(rs.getString("scollege"));
                s.setSbirth(rs.getString("sbirth"));
            }
            rs.close();
            stmt.close();
            conn.close();
        }
        catch (SQLException ex3) {
            ex3.printStackTrace();
        }
        return s;
    }
    //根据性别查询学生信息
    public Student findStudentsex(String ssex) {
    Connection conn = null;                                    //连接数据库对象
    Statement stmt = null;                                     //执行 sql 语句的对象
    ResultSet rs = null;                                       //执行 sql 语句的返回结果
        String sql = "select *from student where ssex='" + ssex + "'";
        Student s = null;
        DBConn db = new DBConn();
        conn = db.getConn();
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            if (rs.next()) {
                s = new Student();
                s.setSno(rs.getString("sno"));
                s.setSname(rs.getString("sname"));
                s.setSsex(rs.getString("ssex"));
                s.setSethnix(rs.getString("sethnix"));
                s.setShome(rs.getString("shome"));
                s.setSyear(rs.getString("syear"));
                s.setSmajor(rs.getString("smajor"));
                s.setScollege(rs.getString("scollege"));
                s.setSbirth(rs.getString("sbirth"));
            }
            rs.close();
            stmt.close();
            conn.close();
        }
```

```
        catch (SQLException ex3) {
            ex3.printStackTrace();
        }
        return s;
    }
    //根据专业查询学生信息
    public Student findStudenmajor(String smajor) {
    Connection conn = null;                                    //连接数据库对象
    Statement stmt = null;                                     //执行 sql 语句的对象
    ResultSet rs = null;                                       //执行 sql 语句的返回结果
        String sql = "select *from student where smajor='" + smajor + "'";
        Student s = null;
        DBConn db = new DBConn();
        conn = db.getConn();
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            if (rs.next()) {
                s = new Student();
                s.setSno(rs.getString("sno"));
                s.setSname(rs.getString("sname"));
                s.setSsex(rs.getString("ssex"));
                s.setSethnix(rs.getString("sethnix"));
                s.setShome(rs.getString("shome"));
                s.setSyear(rs.getString("syear"));
                s.setSmajor(rs.getString("smajor"));
                s.setScollege(rs.getString("scollege"));
                s.setSbirth(rs.getString("sbirth"));
            }
            rs.close();
            stmt.close();
            conn.close();
        }
        catch (SQLException ex3) {
            ex3.printStackTrace();
        }
        return s;
    }
    //按分院查询所有学生信息
    public Vector findStudentcollege(String scollege) {
    Connection conn = null;                                    //连接数据库对象
    Statement stmt = null;                                     //执行 sql 语句的对象
    ResultSet rs = null;                                       //执行 sql 语句的返回结果
        Vector list = new Vector();
        DBConn db = new DBConn();
        conn = db.getConn();
        String sql = "select *from student where scollege='" + scollege + "'";
```

```
        Student s = null;
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            int i = 0;
            while (rs.next()) {
                s = new Student();
                s.setSno(rs.getString("sno"));
                s.setSname(rs.getString("sname"));
                s.setSsex(rs.getString("ssex"));
                s.setSethnix(rs.getString("sethnix"));
                s.setShome(rs.getString("shome"));
                s.setSyear(rs.getString("syear"));
                s.setSmajor(rs.getString("smajor"));
                s.setScollege(rs.getString("scollege"));
                s.setSbirth(rs.getString("sbirth"));
                list.add(i,
                        s.getSno() + " " + s.getSname() + " " + s.getSsex() + " " +
                        s.getSethnix() + " " + s.getShome() + " " + s.getSyear() +
                        " " + s.getSmajor() + " " + s.getScollege() + " " +
                        s.getSbirth());
                i++;
            }
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception em) {
            em.printStackTrace();
            System.out.println("sql 有错");
        }
        return list;
    }
    //获得全部分院名称
    public Vector selectStudentcollege() {
    Connection conn = null;                                    //连接数据库对象
    Statement stmt = null;                                     //执行 sql 语句的对象
    ResultSet rs = null;                                       //执行 sql 语句的返回结果
        DBConn db = new DBConn();
        conn = db.getConn();
        String sql = "select scollege from student";
        Vector v = new Vector();
        try {

            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            while (rs.next()) {
                v.add(rs.getString("scollege"));
```

```
            }
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception em) {
            em.printStackTrace();
            System.out.println("sql 有错");
        }
        return v;
    }
    //获得全部专业
    public Vector selectStudentmajor() {
    Connection conn = null;                                    //连接数据库对象
    Statement stmt = null;                                     //执行 sql 语句的对象
    ResultSet rs = null;                                       //执行 sql 语句的返回结果
        DBConn db = new DBConn();
        conn = db.getConn();
        String sql = "select smajor from student";
        Vector v = new Vector();
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            while (rs.next()) {
                v.add(rs.getString("smajor"));
            }
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception em) {
            em.printStackTrace();
            System.out.println("sql 有错");
        }
        return v;
    }
    //按专业查询
    public Vector findStudentmajor(String smajor) {
    Connection conn = null;                                    //连接数据库对象
    Statement stmt = null;                                     //执行 sql 语句的对象
    ResultSet rs = null;                                       //执行 sql 语句的返回结果
        Vector list = new Vector();
        DBConn db = new DBConn();
        conn = db.getConn();
        String sql = "select *from student where smajor='" + smajor + "'";
        Student s = null;
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
```

```
            int i = 0;
            while (rs.next()) {
                s = new Student();
                s.setSno(rs.getString("sno"));
                s.setSname(rs.getString("sname"));
                s.setSsex(rs.getString("ssex"));
                s.setSethnix(rs.getString("sethnix"));
                s.setShome(rs.getString("shome"));
                s.setSyear(rs.getString("syear"));
                s.setSmajor(rs.getString("smajor"));
                s.setScollege(rs.getString("scollege"));
                s.setSbirth(rs.getString("sbirth"));
                list.add(i,
                        s.getSno() + " " + s.getSname() + " " + s.getSsex() + " " +
                        s.getSethnix() + " " + s.getShome() + " " + s.getSyear() +
                        " " + s.getSmajor() + " " + s.getScollege() + " " +
                        s.getSbirth());
                i++;
            }
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception em) {
            em.printStackTrace();
            System.out.println("sql 有错");
        }
        return list;
    }
    //按性别查询
    public Vector findStudentsex1(String ssex) {
    Connection conn = null;                                    //连接数据库对象
    Statement stmt = null;                                     //执行 sql 语句的对象
    ResultSet rs = null;                                       //执行 sql 语句的返回结果
        Vector list = new Vector();
        DBConn db = new DBConn();
        conn = db.getConn();
        String sql = "select *from student where ssex='" + ssex + "'";
        Student s = null;
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            int i = 0;
            while (rs.next()) {
                s = new Student();
                s.setSno(rs.getString("sno"));
                s.setSname(rs.getString("sname"));
                s.setSsex(rs.getString("ssex"));
```

```
                s.setSethnix(rs.getString("sethnix"));
                s.setShome(rs.getString("shome"));
                s.setSyear(rs.getString("syear"));
                s.setSmajor(rs.getString("smajor"));
                s.setScollege(rs.getString("scollege"));
                s.setSbirth(rs.getString("sbirth"));
                list.add(i,
                        s.getSno() + " " + s.getSname() + " " + s.getSsex() + " " +
                        s.getSethnix() + " " + s.getShome() + " " + s.getSyear() +
                        " " + s.getSmajor() + " " + s.getScollege() + " " +
                        s.getSbirth());
                i++;
            }
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception em) {
            em.printStackTrace();
            System.out.println("sql 有错");
        }
        return list;
    }
//按姓名查询
    public Vector find Student_name(String sname) {
    Connection conn = null;                                    //连接数据库对象
    Statement stmt = null;                                     //执行 sql 语句的对象
    ResultSet rs = null;                                       //执行 sql 语句的返回结果
        Vector list = new Vector();
        DBConn db = new DBConn();
        conn = db.getConn();
        String sql = "select *from student where sname='" + sname + "'";
        Student s = null;
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            int i = 0;
            while (rs.next()) {
                s = new Student();
                s.setSno(rs.getString("sno"));
                s.setSname(rs.getString("sname"));
                s.setSsex(rs.getString("ssex"));
                s.setSethnix(rs.getString("sethnix"));
                s.setShome(rs.getString("shome"));
                s.setSyear(rs.getString("syear"));
                s.setSmajor(rs.getString("smajor"));
                s.setScollege(rs.getString("scollege"));
                s.setSbirth(rs.getString("sbirth"));
```

```
                list.add(i,
                        s.getSno() + " " + s.getSname() + " " + s.getSsex() + " " +
                        s.getSethnix() + " " + s.getShome() + " " + s.getSyear() +
                        " " + s.getSmajor() + " " + s.getScollege() + " " +
                        s.getSbirth());
                i++;
            }
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception em) {
            em.printStackTrace();
            System.out.println("sql 有错");
        }
        return list;
    }
    //查询全部
    public Vector selectStudent() {
    Connection conn = null;                          //连接数据库对象
    Statement stmt = null;                           //执行 sql 语句的对象
    ResultSet rs = null;                             //执行 sql 语句的返回结果
        DBConn db = new DBConn();
        conn = db.getConn();
        String sql = "select sno from student";
        Vector v = new Vector();
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            while (rs.next()) {
                v.add(rs.getString("sno"));
            }
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception em) {
            em.printStackTrace();
            System.out.println("sql 有错");
        }
        return v;
    }
    //通过学号查询全部
    public String findOneStudent(String sno) {
    Connection conn = null;                          //连接数据库对象
    Statement stmt = null;                           //执行 sql 语句的对象
    ResultSet rs = null;                             //执行 sql 语句的返回结果
        DBConn db = new DBConn();
        conn = db.getConn();
```

```
            String sql = "select sname from student where sno='" + sno + "'";
            String s = "";
            try {
                stmt = conn.createStatement();
                rs = stmt.executeQuery(sql);
                if (rs.next()) {
                    s = rs.getString("sname");
                }
                rs.close();
                stmt.close();
                conn.close();
            } catch (Exception em) {
                em.printStackTrace();
                System.out.println("sql 有错");
            }
            return s;
        }
    }
```

2．CourseManager.java

CourseManager 类用于管理 Course 类，主要包含：添加课程信息的方法——addCourse；删除课程信息的方法——deleteCourse；修改课程信息的方法——update_Course；根据课程编号查询课程信息的方法——findCourse；根据课程名称查询课程信息的方法——findCourse_name；根据教师名称查询课程信息的方法——findCourse_teacher；根据专业查询学生信息的方法——findStudenmajor 等方法。

其代码如下：

```
package cvit.com.manager;
import cvit.com.manager.*;
import cvit.com.pub.DBConn;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.Connection;
import cvit.com.model.Course;
import java.sql.SQLException;
import java.util.Vector;
//课程管理类
public class CourseManager {
    //添加课程信息
    public int addCourse(Course c) {
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
        DBConn db = new DBConn();
        conn = db.getConn();
```

```
        int i = 0;
        String sql = "insert into course values('" + c.getCno()+ "','" +
                        c.getCname() + "','" + c.getCteacher() +"','" +
                        c.getCtype() + "','" + c.getCplace() +"','" + c.getCtime() +"')";
        try {
            stmt = conn.createStatement();
            i = stmt.executeUpdate(sql);
            stmt.close();
            conn.close();
        } catch (Exception e1) {
            e1.printStackTrace();
        }
        return i;
    }
    //删除课程信息
     public int deleteCourse(String cno) {
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
        DBConn db = new DBConn();
        conn = db.getConn();
        int i = 0;
        String sql = "delete from course where cno='" + cno +"'";
        try {
            stmt = conn.createStatement();
            i = stmt.executeUpdate(sql);
            stmt.close();
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return i;
    }
    //修改课程信息
    public int update_Course(String cno, String cname, String cteacher,
                                        String ctype, String cplace, String ctime)
{
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
        DBConn db = new DBConn();
        conn = db.getConn();
        int i = 0;
        String sql = "update course set cname='" + cname +"',cteacher='"+
                        cteacher + "',ctype='" + ctype +"',cplace='"+
                        cplace + "',ctime='" + ctime + "' wherecno='" + cno +"'";
        try {
```

```
            stmt = conn.createStatement();
            i = stmt.executeUpdate(sql);
            stmt.close();
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return i;
    }
    //根据课程编号查询课程信息
    public Course findCourse(String cno) {
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
        String sql = "select *from course where cno='" + cno +"'";
        Course c = null;
        DBConn db = new DBConn();
        conn = db.getConn();
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            if (rs.next()) {
                c = new Course();
                c.setCno(rs.getString("cno"));
                c.setCname(rs.getString("cname"));
                c.setCteacher(rs.getString("cteacher"));
                c.setCtype(rs.getString("ctype"));
                c.setCplace(rs.getString("cplace"));
                c.setCtime(rs.getString("ctime"));
            }
            rs.close();
            stmt.close();
            conn.close();
        }
        catch (SQLException ex3) {
            ex3.printStackTrace();
        }
        return c;
    }
    //根据课程名称查询课程信息
    public Course findCourse_name(String cname) {
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
        String sql = "select *from course where cname='" + cname+"'";
        Course c = null;
        DBConn db = new DBConn();
```

```
        conn = db.getConn();
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            if (rs.next()) {
                c = new Course();
                c.setCno(rs.getString("cno"));
                c.setCname(rs.getString("cname"));
                c.setCteacher(rs.getString("cteacher"));
                c.setCtype(rs.getString("ctype"));
                c.setCplace(rs.getString("cplace"));
                c.setCtime(rs.getString("ctime"));
            }
            rs.close();
            stmt.close();
            conn.close();
        }
        catch (SQLException ex3) {
            ex3.printStackTrace();
        }
        return c;
    }
    //根据教师名称查询课程信息
    public Course findCourse_teacher(String cteacher) {
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
        String sql = "select *from course where cteacher='" +cteacher + "'";
        Course c = null;
        DBConn db = new DBConn();
        conn = db.getConn();
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            if (rs.next()) {
                c = new Course();
                c.setCno(rs.getString("cno"));
                c.setCname(rs.getString("cname"));
                c.setCteacher(rs.getString("cteacher"));
                c.setCtype(rs.getString("ctype"));
                c.setCplace(rs.getString("cplace"));
                c.setCtime(rs.getString("ctime"));
            }
            rs.close();
            stmt.close();
            conn.close();
        }
```

```
        catch (SQLException ex3) {
            ex3.printStackTrace();
        }
        return c;
    }
    //查询全部课程名称
    public Vector selectCourse() {
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
        DBConn db = new DBConn();
        conn = db.getConn();
        String sql = "select cname from course";
        Vector v = new Vector();
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            while (rs.next()) {
                v.add(rs.getString("cname"));
            }
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception em) {
            em.printStackTrace();
            System.out.println("sql 有错");
        }
        return v;
    }
    //获得全部教师名称
    public Vector selectCourse2() {
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
        DBConn db = new DBConn();
        conn = db.getConn();
        String sql = "select cteacher from course";
        Vector v = new Vector();
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            while (rs.next()) {
                v.add(rs.getString("cteacher"));
            }
            rs.close();
            stmt.close();
            conn.close();
```

```
        } catch (Exception em) {
            em.printStackTrace();
            System.out.println("sql 有错");
        }
        return v;
    }
    //根据课程名称查询
    public Course findOneCourse(String cname) {
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
        DBConn db = new DBConn();
        conn = db.getConn();
        String sql ="select cno,cteacher,cplace,ctype,ctime from course
                    where cname='" + cname + "'";
        Course c = null;
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            if (rs.next()) {
                c = new Course();
                c.setCno(rs.getString("cno"));
                c.setCteacher(rs.getString("cteacher"));
                c.setCtype(rs.getString("ctype"));
                c.setCplace(rs.getString("cplace"));
                c.setCtime(rs.getString("ctime"));
            }
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception em) {
            em.printStackTrace();
            System.out.println("sql 有错");
        }
        return c;
    }
    //根据教师名称查询
    public Course findOneCourse2(String cteacher) {
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
        DBConn db = new DBConn();
        conn = db.getConn();
        String sql ="select cno,cname,cplace,ctype,ctime from
                        course where cteacher='" +cteacher + "'";
        Course c = null;
        try {
```

```
                stmt = conn.createStatement();
                rs = stmt.executeQuery(sql);
                if (rs.next()) {
                    c = new Course();
                    c.setCno(rs.getString("cno"));
                    c.setCname(rs.getString("cname"));
                    c.setCtype(rs.getString("ctype"));
                    c.setCplace(rs.getString("cplace"));
                    c.setCtime(rs.getString("ctime"));
                }
                rs.close();
                stmt.close();
                conn.close();
            } catch (Exception em) {
                em.printStackTrace();
                System.out.println("sql 有错");
            }
            return c;
        }
    }
```

3．GradeManager.java

GradeManager 类用于管理 Grade 类，主要包含：添加成绩的方法——addGrade，修改成绩的方法——updateGrade，修改课程信息的方法——update_Course，根据学号、课程名称查询学生成绩信息的方法——findGrade，根据学号查询一个学生全部学科的成绩——selectGradestudent 等方法。

其代码如下：

```
package cvit.com.manager;
import cvit.com.model.Grade;
import cvit.com.pub.DBConn;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.Connection;
import cvit.com.model.Grade;
import java.sql.SQLException;
import java.util.Vector;
import cvit.com.model.ZH;
//成绩管理类
public class GradeManager {
    //添加成绩
    public int addGrade(Grade g) {
    Connection conn = null;                          //连接数据库对象
    Statement stmt = null;                           //执行 sql 语句的对象
    ResultSet rs = null;                             //执行 sql 语句的返回结果
        DBConn db = new DBConn();
```

```
        conn = db.getConn();
        int i = 0;
        String sql = "update grade set grade='" + g.getGrade() + "'where sno='" +
                        g.getSno() + "'and cno='" + g.getCno() + "'";
        try {
            stmt = conn.createStatement();
            i = stmt.executeUpdate(sql);
            stmt.close();
            conn.close();
        } catch (Exception e1) {
            e1.printStackTrace();
        }
        return i;       }
    //选课
    public int chooseCourse(String sno, String cno) {
    Connection conn = null;                                    //连接数据库对象
    Statement stmt = null;                                     //执行 sql 语句的对象
    ResultSet rs = null;                                       //执行 sql 语句的返回结果
        DBConn db = new DBConn();
        conn = db.getConn();
        int i = 0;
        String sql = "insert into grade values('" + sno + "','" + cno +"',' ')";
        try {
            stmt = conn.createStatement();
            i = stmt.executeUpdate(sql);
            stmt.close();
            conn.close();
        } catch (Exception e1) {
            e1.printStackTrace();
        }
        return i;
    }
    //根据学号、课程名查询一个学生成绩
    public Grade findGrade(String sno, String cno) {
    Connection conn = null;                                    //连接数据库对象
    Statement stmt = null;                                     //执行 sql 语句的对象
    ResultSet rs = null;                                       //执行 sql 语句的返回结果
        String sql = "select *from grade where sno='" + sno + "'and cno='" + cno + "'";
        Grade g = null;
        DBConn db = new DBConn();
        conn = db.getConn();
        try {
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            if (rs.next()) {
                g = new Grade();
                g.setSno(rs.getString("sno"));
```

```
                g.setCno(rs.getString("cno"));
                g.setGrade(rs.getString("grade"));
            }
            rs.close();
            stmt.close();
            conn.close();
        }
        catch (SQLException ex3) {
            ex3.printStackTrace();
        }
        return g;
    }
    //修改成绩
    public int updateGrade(String sno, String cno, String grade) {
    Connection conn = null;                                    //连接数据库对象
    Statement stmt = null;                                     //执行 sql 语句的对象
    ResultSet rs = null;                                       //执行 sql 语句的返回结果
        DBConn db = new DBConn();
        conn = db.getConn();
        int i = 0;
        String sql = "update grade set cno='" + cno + "',grade='" + grade +
                    "' where sno='" + sno + "'";
        try {
            stmt = conn.createStatement();
            i = stmt.executeUpdate(sql);
            stmt.close();
            conn.close();
        } catch (Exception e1) {
            e1.printStackTrace();
        }
        return i;
    }
    //查询一个学生全部学科的成绩
    public Vector selectGradestudent(String sno) {
            Vector list = new Vector();
            DBConn db = new DBConn();
            conn = db.getConn();
            String sql = "select student.sno,sname,course.cno,cname,grade from
                        student,course,grade where grade.sno='" + sno + "'";
            ZH zh = null;
            try {
                stmt = conn.createStatement();
                rs = stmt.executeQuery(sql);
                int i = 0;
                while (rs.next()) {
                    zh = new ZH();
                    zh.setSno(rs.getString("sno"));
```

```
                zh.setSname(rs.getString("sname"));
                zh.setCno(rs.getString("cno"));
                zh.setCname(rs.getString("cname"));
                zh.setGrade(rs.getString("grade"));
                list.add(i,zh.getSno() + " " + zh.getSname() + " " + zh.getCno() + " "+
                        zh.getCname() + " " + zh.getGrade() + " ");
                i++;
            }
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception em) {
            em.printStackTrace();
            System.out.println("sql 有错");
        }
        return list;
    }
}
```

5.4.6 视图层设计与实现

视图层是用户看到并与之交互的界面，对于用户来说，视图层就是整个系统，因为其他层对于用户是透明的。根据功能模块的划分与以用户为中心的原则，共设计 18 个类，放置在 cvit.com.view 包中，用于满足视图层需求，其中包括：StuMainFrame——主界面，添加学生信息界面——Add_studentPanel，修改学生信息界面——Update_studentPanel，删除学生信息界面——Delete_studentPanel 等界面，下面一一说明。

1．StuMainFrame.java

StuMainFrame 类用于实现主界面功能，且是学生信息管理系统的主运行类，类中的 Main 方法是程序的运行入口，其主界面如图 5-6 所示。

图 5-6　学生管理系统主界面

其菜单如图 5-7 至图 5-12 所示。

图 5-7　学生管理模块菜单

图 5-8　课程管理模块菜单

图 5-9　成绩管理模块菜单

图 5-10　信息查询之学生查询模块菜单

图 5-11　信息查询之课程查询模块菜单

图 5-12　信息查询之成绩查询模块菜单

其代码如下：

```
package cvit.com.view;
import java.awt.*;
import java.awt.BorderLayout;
import javax.swing.JFrame;
import javax.swing.*;
import java.awt.event.*;
import java.awt.Dimension;
import com.sun.media.sound.Toolkit.*;
import java.awt.Font;
import cvit.com.view.*;
//系统主界面
public class StuMainFrame extends JFrame implements ActionListener {
    //定义菜单条
    JMenuBar mb = new JMenuBar();
    //定义菜单
    JMenu systemment = new JMenu("系统管理");
    JMenu studentment = new JMenu("学生管理");
    JMenu coursement = new JMenu("课程管理");
    JMenu gradement = new JMenu("成绩管理");
    JMenu informationment = new JMenu("信息查询");
    JMenu student_select = new JMenu("学生查询");
    JMenu course_select = new JMenu("课程查询");
    JMenu grade_select = new JMenu("成绩查询");
    JMenuItem exit = new JMenuItem("退出");
    JMenuItem student_add = new JMenuItem("增加");
    JMenuItem student_update = new JMenuItem("修改");
    JMenuItem student_delete = new JMenuItem("删除");
    JMenuItem student_choose = new JMenuItem("学生选课");
    JMenuItem course_add = new JMenuItem("课程增加");
```

```
JMenuItem course_update = new JMenuItem("课程修改");
JMenuItem course_delete = new JMenuItem("课程删除");
JMenuItem grade_add = new JMenuItem("成绩增加");
JMenuItem grade_update = new JMenuItem("成绩修改");
JMenuItem select_no = new JMenuItem("按学号查询");
JMenuItem select_name = new JMenuItem("按学生姓名查询");
JMenuItem select_sex = new JMenuItem("按学生性别查询");
JMenuItem select_major = new JMenuItem("按专业查询");
JMenuItem select_college = new JMenuItem("按学院查询");
JMenuItem select_cname = new JMenuItem("按课程名称查询");
JMenuItem select_cteacher = new JMenuItem("按授课教师查询");
JMenuItem select_allgrade = new JMenuItem("查询所有科目成绩");
JPanel p = null;
JTextField jtf1 = new JTextField();
BorderLayout borderLayout1 = new BorderLayout();
public StuMainFrame() {
    try {
        jbInit();
    } catch (Exception exception) {
        exception.printStackTrace();
    }
}

private void jbInit() throws Exception {
    getContentPane().setLayout(borderLayout1);
    this.getContentPane().setBackground(new Color(168, 233, 216));
    this.setJMenuBar(mb);
    jtf1.setBackground(SystemColor.control);
    jtf1.setEnabled(false);
    jtf1.setFont(new java.awt.Font("宋体", Font.BOLD, 12));
    jtf1.setForeground(Color.cyan);
    jtf1.setSelectionColor(Color.black);
    //以下是向菜单条添加菜单项
    mb.add(systemment);
    mb.add(studentment);
    mb.add(coursement);
    mb.add(gradement);
    mb.add(informationment);
    //以下是向菜单中添加菜单项
    systemment.add(exit); //系统管理退出
    studentment.add(student_add);                    //学生管理
    studentment.add(student_update);
    studentment.add(student_delete);
    studentment.add(student_choose);
    coursement.add(course_add);                      //课程管理
    coursement.add(course_update);
    coursement.add(course_delete);
```

```
gradement.add(grade_add);                    //成绩管理
gradement.add(grade_update);
informationment.add(student_select);         //信息查询
informationment.add(course_select);
informationment.add(grade_select);
//以下是向二级菜单中添加菜单项
student_select.add(select_no);
student_select.add(select_name);
student_select.add(select_sex);
student_select.add(select_major);
student_select.add(select_college);
course_select.add(select_cname);
course_select.add(select_cteacher);
grade_select.add(select_allgrade);
this.getContentPane().add(jtf1, java.awt.BorderLayout.SOUTH);
exit.addActionListener(this);
//注册菜单监听
student_add.addActionListener(this);         //添加学生信息
student_update.addActionListener(this);      //修改学生信息
student_delete.addActionListener(this);      //删除学生信息
student_choose.addActionListener(this);      //学生选课
course_add.addActionListener(this);          //添加课程信息
course_update.addActionListener(this);       //修改课程信息
course_delete.addActionListener(this);       //删除学生信息
grade_add.addActionListener(this);           //添加成绩
grade_update.addActionListener(this);        //修改成绩
select_no.addActionListener(this);           //按学号查询
select_major.addActionListener(this);        //按专业查询
select_cname.addActionListener(this);        //按课程名称查询
select_cteacher.addActionListener(this);     //按任课教师查询
select_college.addActionListener(this);      //按学院查询
select_sex.addActionListener(this);          //按性别查询
select_name.addActionListener(this);         //按姓名查询
select_allgrade.addActionListener(this);     //所有成绩
//版权信息
jtf1.setText("学生信息管理系统版由 CVIT 开发,盗版必究! ");
jtf1.setHorizontalAlignment(SwingConstants.CENTER);
this.setTitle("学生信息管理系统");            //窗体名称
this.setSize(600, 600);                      //大小
this.setVisible(true);                       //可见性
//以下为设置窗体居中
//获得屏幕的宽和高
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
//获得当前窗体的宽和高
Dimension frameSize = this.getSize();
//设置窗体居中
if (frameSize.height > screenSize.height) {
```

```
            frameSize.height = screenSize.height;
        }
        if (frameSize.width > screenSize.width) {
            frameSize.width = screenSize.width;
        }
        this.setLocation((screenSize.width−frameSize.width) / 2,
                         (screenSize.height−frameSize.height) / 2);
        this.setVisible(true);       }

    public static void main(String[] args) {
        StuMainFrame sm = new StuMainFrame();
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == exit) {                              //退出
            this.setVisible(false);
        }
        //添加学生信息
        if (e.getSource() == student_add) {
            if (p != null) {
                this.remove(p);
            }
            //定义一个添加学生信息页面的对象
            p = new Add_studentPanel();
            //将 panel 放在页面的中间
          this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
            this.setVisible(true);
        }
        //修改学生信息
        if (e.getSource() == student_update) {
            if (p != null) {
                this.remove(p);
            }
            p = new Update_studentPanel();
            this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
            this.setVisible(true);
        }
        //删除学生信息
        if (e.getSource() == student_delete) {
            if (p != null) {
                this.remove(p);
            }
            p = new Delete_studentPanel();
            this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
            this.setVisible(true);
        }
        //学生选课
        if (e.getSource() == student_choose) {
```

```
            if (p != null) {
                this.remove(p);
            }
            p = new Student_choosePanel();
            this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
            this.setVisible(true);
        }
        //添加课程信息
        if (e.getSource() == course_add) {
            if (p != null) {
                this.remove(p);
            }
            p = new Add_coursePanel();
            this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
            this.setVisible(true);
        }
        //修改课程信息
        if (e.getSource() == course_update) {
            if (p != null) {
                this.remove(p);
            }
            p = new Update_coursePanel();
            this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
            this.setVisible(true);
        }
        //删除课程信息
        if (e.getSource() == course_delete) {
            if (p != null) {
                this.remove(p);
            }
            p = new Delete_coursePanel();
            this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
            this.setVisible(true);
        }
        //成绩添加
        if (e.getSource() == grade_add) {
            if (p != null) {
                this.remove(p);
            }
            p = new Add_gradePanel();
            this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
            this.setVisible(true);
        }
        //成绩修改
        if (e.getSource() == grade_update) {
            if (p != null) {
                this.remove(p);
```

```
        }
        p = new Update_gradePanel();
        this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
        this.setVisible(true);
    }
    //按学号查询
    if (e.getSource() == select_no) {
        if (p != null) {
            this.remove(p);
        }
        p = new Select_studentnoPanel1();
        this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
        this.setVisible(true);
    }
    //按分院查询
    if (e.getSource() == select_college) {
        if (p != null) {
            this.remove(p);
        }
        p = new Select_studentcollegePanel();
        this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
        this.setVisible(true);
    }
    //按专业查询
    if (e.getSource() == select_major) {
        if (p != null) {
            this.remove(p);
        }
        p = new Select_studentmajorPanel1();
        this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
        this.setVisible(true);
    }
    //按课程名称查询
    if (e.getSource() == select_cname) {
        if (p != null) {
            this.remove(p);
        }
        p = new Select_coursecnamePanel();
        this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
        this.setVisible(true);
    }
    //按任课教师查询
    if (e.getSource() == select_cteacher) {
        if (p != null) {
            this.remove(p);
        }
        p = new Select_coursecteacherPanel1();
```

```
                this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
                this.setVisible(true);
            }
            //按学生姓名查询
            if (e.getSource() == select_name) {
                if (p != null) {
                    this.remove(p);
                }
                p = new Select_studentnamePanel();
                this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
                this.setVisible(true);
            }
            //按性别查询
            if (e.getSource() == select_sex) {
                if (p != null) {
                    this.remove(p);
                }
                p = new Select_studentsexPanel();
                this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
                this.setVisible(true);
            }
           //查询所有科目成绩
            if (e.getSource() == select_allgrade) {
                if (p != null) {
                    this.remove(p);
                }
                p = new Select_gradestudentPanel1();
                this.getContentPane().add(p, java.awt.BorderLayout.CENTER);
                this.setVisible(true);
            }
        }
```

2．Add_studentPanel.java

Add_studentPanel 类用于实现添加学生信息界面功能，其界面如图 5-13 所示。

图 5-13　添加学生信息界面

其代码如下：

```
package cvit.com.view;
import cvit.com.manager.StudentManager;
import cvit.com.model.Student;
import javax.swing.JPanel;
import javax.swing.JButton;
import java.awt.Rectangle;
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JOptionPane;
import javax.swing.JComboBox;
//添加学生信息
public class Add_studentPanel extends JPanel {
    public Add_studentPanel() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
        this.setSize(600, 600);
        this.setBackground(new Color(168, 233, 216));
        this.setVisible(true);
        jLabel1.setText("学号");
        jLabel1.setBounds(new Rectangle(15, 36, 42, 15));
```

```
jLabel2.setText("性别");
jLabel2.setBounds(new Rectangle(14, 74, 42, 15));
tsno.setBounds(new Rectangle(79, 33, 77, 20));
jLabel3.setText("姓名");
jLabel3.setBounds(new Rectangle(187, 35, 42, 15));
jLabel4.setText("民族");
jLabel4.setBounds(new Rectangle(184, 75, 42, 15));
tsname.setBounds(new Rectangle(250, 32, 77, 20));
jLabel5.setText("出生日期");
jLabel5.setBounds(new Rectangle(5, 112, 67, 15));
jLabel6.setText("入学时间");
jLabel6.setBounds(new Rectangle(185, 112, 57, 15));
tsbirth.setBounds(new Rectangle(79, 111, 77, 20));
tstime.setBounds(new Rectangle(251, 111, 77, 20));
jLabel7.setText("学院");
jLabel7.setBounds(new Rectangle(15, 154, 42, 15));
tscollege.setBounds(new Rectangle(79, 153, 77, 20));
jLabel8.setText("专业");
jLabel8.setBounds(new Rectangle(188, 154, 42, 15));
tsmajor.setBounds(new Rectangle(251, 156, 77, 20));
jLabel9.setText("籍贯");
jLabel9.setBounds(new Rectangle(16, 197, 42, 15));
tsplace.setBounds(new Rectangle(77, 198, 251, 20));
btn1.setBounds(new Rectangle(35, 247, 81, 23));
btn1.setText("添加");
btn1.addActionListener(new Add_studentPanel_btn1_actionAdapter(this));
btn2.setBounds(new Rectangle(151, 247, 81, 23));
btn2.setText("清空");
btn2.addActionListener(new Add_studentPanel_btn2_actionAdapter(this));
btn3.setBounds(new Rectangle(265, 247, 81, 23));
btn3.setText("退出");
btn3.addActionListener(new Add_studentPanel_btn3_actionAdapter(this));
jcb_sex.setBounds(new Rectangle(77, 66, 43, 23));
jcb_ethnix.setBounds(new Rectangle(253, 66, 58, 23));
this.add(tsno);
this.add(jLabel3);
this.add(tsname);
this.add(btn1);
this.add(btn2);
this.add(jLabel5);
this.add(jLabel1);
this.add(tscollege);
this.add(tsbirth);
this.add(tstime);
this.add(jLabel6);
this.add(jLabel4);
this.add(jLabel2);
```

```
            this.add(jLabel7);
            this.add(jLabel8);
            this.add(tsmajor);
            this.add(jLabel9);
            this.add(tsplace);
            this.add(btn3);
            this.add(jcb_ethnix);
            this.add(jcb_sex);
            jcb_sex.addItem("男");                    //向性别的列表中添加内容
            jcb_sex.addItem("女");
            jcb_ethnix.addItem("汉族");               //向民族的列表中添加内容
            jcb_ethnix.addItem("蒙族");
            jcb_ethnix.addItem("回族");
            jcb_ethnix.addItem("满族");
        }
        JLabel jLabel1 = new JLabel();
        JLabel jLabel2 = new JLabel();
        JTextField tsno = new JTextField();
        JLabel jLabel3 = new JLabel();
        JLabel jLabel4 = new JLabel();
        JTextField tsname = new JTextField();
        JLabel jLabel5 = new JLabel();
        JLabel jLabel6 = new JLabel();
        JTextField tsbirth = new JTextField();
        JTextField tstime = new JTextField();
        JLabel jLabel7 = new JLabel();
        JTextField tscollege = new JTextField();
        JLabel jLabel8 = new JLabel();
        JTextField tsmajor = new JTextField();
        JLabel jLabel9 = new JLabel();
        JTextField tsplace = new JTextField();
        JButton btn1 = new JButton();
        JButton btn2 = new JButton();
        JButton btn3 = new JButton();
        JComboBox jcb_sex = new JComboBox();
        JComboBox jcb_ethnix = new JComboBox();
    public void btn1_actionPerformed(ActionEvent actionEvent) {
    //定义空的学生信息属性
        String sno = "";
        String sname = "";
        String ssex = "";
        String sbirth = "";
        String syear = "";
        String shome = "";
            String smajor = "";
            String scollege = "";
            if (!tsno.getText().equals("")) {
```

```
            sno = tsno.getText();
        } else {
            //弹出对话框信息
            JOptionPane.showMessageDialog(this, "学号不能为空！", "提示信息",
                                                    JOptionPane.ERROR_MESSAGE);
            return;
        }
        if (!tsname.getText().equals("")) {
            sname = tsname.getText();
        } else {
            JOptionPane.showMessageDialog(this, "姓名不能为空！", "提示信息",
                                                    JOptionPane.ERROR_MESSAGE);
            return;
        }
        if (!tsbirth.getText().equals("")) {
            sbirth = tsbirth.getText();
        } else {
            JOptionPane.showMessageDialog(this, "出生日期不能为空！", "提示信息",
                                                    JOptionPane.ERROR_MESSAGE);
            return;
        }
        if (!tstime.getText().equals("")) {
            syear = tstime.getText();
        } else {
            JOptionPane.showMessageDialog(this, "入学时间不能为空！", "提示信息",
                                                    JOptionPane.ERROR_MESSAGE);
            return;
        }
        if (!tscollege.getText().equals("")) {
            scollege = tscollege.getText();
        } else {
            JOptionPane.showMessageDialog(this, "学院不能为空！", "提示信息",
                                                    JOptionPane.ERROR_MESSAGE);
            return;
        }
        if (!tsmajor.getText().equals("")) {
            smajor = tsmajor.getText();
        } else {
            JOptionPane.showMessageDialog(this, "专业不能为空！", "提示信息",
                                                    JOptionPane.ERROR_MESSAGE);
            return;
        }
        if (!tsplace.getText().equals("")) {
            shome = tsplace.getText();
        } else {
            JOptionPane.showMessageDialog(this, "籍贯不能为空！", "提示信息",
                                                    JOptionPane.ERROR_MESSAGE);
```

```
                return;
            }
            ssex = jcb_sex.getSelectedItem().toString();              //获得列表中选定的性别信息
            String sethnix = jcb_ethnix.getSelectedItem().toString(); //获得民族列表总选到的信息
            Student s = new Student();                                //定义学生对象 s 进行封装学生信息
            s.setSno(sno);
            s.setSname(sname);
            s.setSsex(ssex);
            s.setSethnix(sethnix);
            s.setShome(shome);
            s.setSyear(syear);
            s.setSmajor(smajor);
            s.setScollege(scollege);
            s.setSbirth(sbirth);
            StudentManager sm = new StudentManager();                 //定义学生管理对象 sm
            int i = sm.addStudent(s);                                 //调用添加学生信息方法
            if (i > 0) {
                JOptionPane.showMessageDialog(this, "添加成功！", "提示信息",
                                                    JOptionPane.INFORMATION_MESSAGE);
                tsno.setText("");
                tsname.setText("");
                tsplace.setText("");
                tstime.setText("");
                tsmajor.setText("");
                tscollege.setText("");
                tsbirth.setText("");
            } else {
                JOptionPane.showMessageDialog(this, "添加失败！", "提示信息",
                                                    JOptionPane.ERROR_MESSAGE);
            }
        }
        //退出按钮关闭当前页面
        public void btn3_actionPerformed(ActionEvent e) {
            this.setVisible(false);
        }
        //清空按钮清空各文本框中的内容
        public void btn2_actionPerformed(ActionEvent e) {
            tsno.setText("");
            tsname.setText("");
            tsplace.setText("");
            tstime.setText("");
            tsmajor.setText("");
            tscollege.setText("");
            tsbirth.setText("");
        }
    }
    //监听
```

```
class Add_studentPanel_btn2_actionAdapter implements ActionListener {
    private Add_studentPanel adaptee;                    //编码时双击添加按钮自动生成代码
    Add_studentPanel_btn2_actionAdapter(Add_studentPanel adaptee) {
        this.adaptee = adaptee;
    }
    public void actionPerformed(ActionEvent e) {
        adaptee.btn2_actionPerformed(e);
    }
}
class Add_studentPanel_btn3_actionAdapter implements ActionListener {
    private Add_studentPanel adaptee;
    Add_studentPanel_btn3_actionAdapter(Add_studentPanel adaptee) {
        this.adaptee = adaptee;
    }
    public void actionPerformed(ActionEvent e) {
        adaptee.btn3_actionPerformed(e);
    }
}
class Add_studentPanel_btn1_actionAdapter implements ActionListener {
    private Add_studentPanel adaptee;
    Add_studentPanel_btn1_actionAdapter(Add_studentPanel adaptee) {
        this.adaptee = adaptee;
    }
    public void actionPerformed(ActionEvent actionEvent) {
        adaptee.btn1_actionPerformed(actionEvent);
    }
}
```

3．Update_studentPanel.java

Update_studentPanel 类用于实现修改学生信息界面功能，其界面如图 5-14 所示。

图 5-14　修改学生信息界面

其代码如下：

```
package cvit.com.view;

import javax.swing.JPanel;
import javax.swing.JLabel;
import java.awt.Rectangle;
import javax.swing.JTextField;
import java.awt.Color;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import cvit.com.manager.StudentManager;
import cvit.com.model.Student;
import javax.swing.JOptionPane;
//修改学生信息
public class Update_studentPanel extends JPanel {
	public Update_studentPanel() {
		try {
			jbInit();
		} catch (Exception exception) {
			exception.printStackTrace();
		}
	}
	private void jbInit() throws Exception {
		this.setLayout(null);
		jLabel1.setText("专业");
		jLabel1.setBounds(new Rectangle(188, 154, 42, 15));
		jLabel2.setText("民族");
		jLabel2.setBounds(new Rectangle(184, 75, 42, 15));
		tsno.setBounds(new Rectangle(79, 33, 77, 20));
		tscollege.setEnabled(false);
		tscollege.setBounds(new Rectangle(79, 153, 77, 20));
		jLabel3.setText("学号");
		jLabel3.setBounds(new Rectangle(15, 36, 42, 15));
		jLabel4.setText("姓名");
		jLabel4.setBounds(new Rectangle(187, 35, 42, 15));
		jLabel5.setText("学院");
		jLabel5.setBounds(new Rectangle(15, 154, 42, 15));
		tsmajor.setEnabled(false);
		tsmajor.setBounds(new Rectangle(251, 156, 77, 20));
		tsname.setEnabled(false);
		tsname.setBounds(new Rectangle(250, 32, 77, 20));
		jLabel6.setText("性别");
		jLabel6.setBounds(new Rectangle(14, 74, 42, 15));
		tstime.setEnabled(false);
		tstime.setBounds(new Rectangle(251, 111, 77, 20));
		tssex.setEnabled(false);
		tssex.setBounds(new Rectangle(77, 73, 77, 20));
```

```
        jLabel7.setText("入学时间");
        jLabel7.setBounds(new Rectangle(175, 112, 67, 15));
        tsbirth.setEnabled(false);
        tsbirth.setBounds(new Rectangle(79, 111, 77, 20));
        tshome.setEnabled(false);
        tshome.setBounds(new Rectangle(77, 198, 251, 20));
        jLabel8.setText("籍贯");
        jLabel8.setBounds(new Rectangle(16, 197, 42, 15));
        tsethnix.setEnabled(false);
        tsethnix.setBounds(new Rectangle(251, 73, 77, 20));
        jLabel9.setText("出生日期");
        jLabel9.setBounds(new Rectangle(12, 112, 53, 15));
        this.setBackground(new Color(168, 233, 216));
        btn_select.setBounds(new Rectangle(30, 241, 81, 23));
        btn_select.setText("查找");
        btn_select.addActionListener(newUpdate_studentPanel_
                                     btn_select_actionAdapter(this));
        btn_update.setBounds(new Rectangle(140, 240, 81, 23));
        btn_update.setText("修改");
        btn_update.addActionListener(new
                                Update_studentPanel_btn_update_actionAdapter(this));
        btn_exit.setBounds(new Rectangle(267, 239, 81, 23));
        btn_exit.setText("退出");
        btn_exit.addActionListener(new
                                 Update_studentPanel_btn_exit_actionAdapter(this));
        this.add(jLabel1);
        this.add(jLabel2);
        this.add(tsno);
        this.add(tscollege);
        this.add(jLabel3);
        this.add(jLabel4);
        this.add(jLabel5);
        this.add(tsmajor);
        this.add(tsname);
        this.add(jLabel6);
        this.add(tstime);
        this.add(tssex);
        this.add(jLabel7);
        this.add(tsbirth);
        this.add(tshome);
        this.add(jLabel8);
        this.add(tsethnix);
        this.add(jLabel9);
        this.add(btn_select);
        this.add(btn_update);
        this.add(btn_exit);
        this.setVisible(true);                                  //设置窗体可见性
```

```
        this.setSize(600, 600);                                    //设置窗体大小
    }
    JLabel jLabel1 = new JLabel();
    JLabel jLabel2 = new JLabel();
    JTextField tsno = new JTextField();
    JTextField tscollege = new JTextField();
    JLabel jLabel3 = new JLabel();
    JLabel jLabel4 = new JLabel();
    JLabel jLabel5 = new JLabel();
    JTextField tsmajor = new JTextField();
    JTextField tsname = new JTextField();
    JLabel jLabel6 = new JLabel();
    JTextField tstime = new JTextField();
    JTextField tssex = new JTextField();
    JLabel jLabel7 = new JLabel();
    JTextField tsbirth = new JTextField();
    JTextField tshome = new JTextField();
    JLabel jLabel8 = new JLabel();
    JTextField tsethnix = new JTextField();
    JLabel jLabel9 = new JLabel();
    JButton btn_select = new JButton();
    JButton btn_update = new JButton();
    JButton btn_exit = new JButton();
    public void btn_select_actionPerformed(ActionEvent e) {
        String sno = tsno.getText().trim();                        //获得学号文本框中的内容
        StudentManager sm = new StudentManager();                  //定义学生管理的对象 sm
        Student s = sm.findStudent(sno);                           //调用学生管理中的按学号查询方法
        //对话框
        if (s == null) {
            JOptionPane.showMessageDialog(this, "没有该用同学！", "提示信息",
                                          JOptionPane.INFORMATION_MESSAGE);
            tsno.setText("");                                      //清空文本框
        } else {
            //设置文本框是否可用 false 为不可用、true 为可用
            tsno.setEnabled(false);
            tsname.setEnabled(true);
            tssex.setEnabled(true);
            tsethnix.setEnabled(true);
            tshome.setEnabled(true);
            tstime.setEnabled(true);
            tsmajor.setEnabled(true);
            tscollege.setEnabled(true);
            tsbirth.setEnabled(true);
            tsno.setText(s.getSno());
            tsname.setText(s.getSname());
            tssex.setText(s.getSsex());
            tsethnix.setText(s.getSethnix());
```

```
                tshome.setText(s.getShome());
                tstime.setText(s.getSyear());
                tsmajor.setText(s.getSmajor());
                tscollege.setText(s.getScollege());
                tsbirth.setText(s.getSbirth());
            }
        }
        public void btn_update_actionPerformed(ActionEvent e) {
            String sno = tsno.getText();                                    //获得学号文本框中的内容作为学号
            String sname = tsname.getText();
            String ssex = tssex.getText();
            String sethnix = tsethnix.getText();
            String shome = tshome.getText();
            String syear = tstime.getText();
            String smajor = tsmajor.getText();
            String scollege = tscollege.getText();
            String sbirth = tsbirth.getText();
            StudentManager sm = new StudentManager();                       //定义学生管理对象 sm
        int i = sm.update_Student(sno, sname, ssex, sethnix, shome, syear,  //调用修改学生信息方法
                                        smajor, scollege, sbirth);
            if (i > 0) {
                JOptionPane.showMessageDialog(this, "修改成功！", "提示信息",
                                                    JOptionPane.INFORMATION_MESSAGE);
                tsno.setEnabled(true);                                      //设置文本框的可用性
                tsname.setEnabled(false);
                tssex.setEnabled(false);
                tsethnix.setEnabled(false);
                tshome.setEnabled(false);
                tstime.setEnabled(false);
                tsmajor.setEnabled(false);
                tscollege.setEnabled(false);
                tsbirth.setEnabled(false);
                tsno.setText("");                                           //清空文本框内容
                tsname.setText("");
                tssex.setText("");
                tsethnix.setText("");
                tshome.setText("");
                tstime.setText("");
                tsmajor.setText("");
                tscollege.setText("");
                tsbirth.setText("");
            } else {
                JOptionPane.showMessageDialog(this, "修改失败！", "提示信息",
                                                    JOptionPane.ERROR_MESSAGE);
            }
        }
        //退出按钮关闭窗体
```

```
        public void btn_exit_actionPerformed(ActionEvent e) {
            this.setVisible(false);
        }
    }
    class Update_studentPanel_btn_exit_actionAdapter implements ActionListener {
        private Update_studentPanel adaptee;
        Update_studentPanel_btn_exit_actionAdapter(Update_studentPanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_exit_actionPerformed(e);
        }
    }
    class Update_studentPanel_btn_update_actionAdapter implements ActionListener {
        private Update_studentPanel adaptee;
        Update_studentPanel_btn_update_actionAdapter(Update_studentPanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_update_actionPerformed(e);
        }
    }
    class Update_studentPanel_btn_select_actionAdapter implements ActionListener {
        private Update_studentPanel adaptee;
        Update_studentPanel_btn_select_actionAdapter(Update_studentPanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_select_actionPerformed(e);
        }
    }
```

4．Delete_studentPanel.java

Delete_studentPanel 类用于实现删除学生信息界面功能，其界面如图 5-15 所示。

图 5-15　删除学生信息界面

其代码如下：

```
package cvit.com.view;
import cvit.com.manager.StudentManager;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JTextField;
import java.awt.Rectangle;
import javax.swing.JLabel;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JOptionPane;
//删除学生信息
public class Delete_studentPanel extends JPanel {
    public Delete_studentPanel() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
        this.setBackground(new Color(168, 233, 216));
        tsno.setBounds(new Rectangle(78, 76, 130, 20));
        jLabel1.setText("学号");
        jLabel1.setBounds(new Rectangle(32, 81, 42, 15));
        btn_delete.setBounds(new Rectangle(235, 76, 81, 23));
        btn_delete.setText("删除");
        btn_delete.addActionListener(new
                                     elete_studentPanel_btn_delete_actionAdapter(this));
        btn_exit.setBounds(new Rectangle(233, 111, 81, 23));
        btn_exit.setText("退出");
        btn_exit.addActionListener(new
                                   Delete_studentPanel_btn_exit_actionAdapter(this));
        this.add(tsno);
        this.add(jLabel1);
        this.add(btn_delete);
        this.add(btn_exit);
        this.setVisible(true);                          //可见性
        this.setSize(600, 600);                         //窗体大小
    }
    JTextField tsno = new JTextField();
    JLabel jLabel1 = new JLabel();
    JButton btn_delete = new JButton();
    JButton btn_exit = new JButton();
```

```
        public void btn_delete_actionPerformed(ActionEvent e) {
            String sno = tsno.getText();                    //获得学号文本框中的学号
            StudentManager sm = new StudentManager();   //定义学生管理对象 sm
            int ok = JOptionPane.showConfirmDialog(this,"是否删除" + tsno.getText() + "同学？
                                                    ","提示信息",
    JOptionPane.OK_CANCEL_OPTION);
            if (ok == JOptionPane.OK_OPTION) {
                int i = sm.deleteUser(sno);                 //调用删除学生方法
                if (i > 0) {
                    JOptionPane.showMessageDialog(this, "删除成功！", "提示信息",
                                                JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(this, "你要删除的同学不存在！",
                                    "提示信息", JOptionPane.ERROR_MESSAGE);
                }
                tsno.setText("");
            }
        }
        //退出按钮关闭页面
        public void btn_exit_actionPerformed(ActionEvent e) {
            this.setVisible(false);
        }
    }
    class Delete_studentPanel_btn_exit_actionAdapter implements ActionListener {
        private Delete_studentPanel adaptee;
        Delete_studentPanel_btn_exit_actionAdapter(Delete_studentPanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_exit_actionPerformed(e);
        }
    }
    class Delete_studentPanel_btn_delete_actionAdapter implements ActionListener {
        private Delete_studentPanel adaptee;
        Delete_studentPanel_btn_delete_actionAdapter(Delete_studentPanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_delete_actionPerformed(e);
        }
    }
```

5．Student_choosePanel.java

Student_choosePanel 类用于实现学生选课界面功能，其界面如图 5-16 所示。

图 5-16　学生选课界面

其代码如下：

```
package cvit.com.view;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JTextField;
import java.awt.Rectangle;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import cvit.com.manager.StudentManager;
import java.util.Vector;
import cvit.com.manager.CourseManager;
import cvit.com.model.Course;
import javax.swing.JOptionPane;
import cvit.com.manager.GradeManager;
//学生选课
public class Student_choosePanel extends JPanel {
    public Student_choosePanel() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
        this.setBackground(new Color(168, 233, 216));
        tsname.setEnabled(false);
```

```
tsname.setBounds(new Rectangle(241, 25, 103, 27));
jcb_course.setBounds(new Rectangle(63, 78, 102, 26));
jcb_course.addItemListener(new
                                Student_choosePanel_jcb_course_itemAdapter(this));
jLabel1.setText("学　号");
jLabel1.setBounds(new Rectangle(11, 34, 42, 15));
jLabel2.setText("姓　名");
jLabel2.setBounds(new Rectangle(176, 34, 42, 15));
jcb_sno.setBounds(new Rectangle(61, 28, 108, 26));
jcb_sno.addItemListener(new Student_choosePanel_jcb_sno_itemAdapter(this));
tcno.setEnabled(false);
tcno.setBounds(new Rectangle(239, 82, 98, 20));
jLabel3.setText("选择课程");
jLabel3.setBounds(new Rectangle(6, 87, 55, 15));
jLabel4.setText("课程编码");
jLabel4.setBounds(new Rectangle(174, 84, 58, 15));
jLabel5.setText("授课教师");
jLabel5.setBounds(new Rectangle(9, 141, 56, 15));
tcteacher.setEnabled(false);
tcteacher.setBounds(new Rectangle(80, 136, 77, 20));
jLabel6.setText("授课地点");
jLabel6.setBounds(new Rectangle(173, 143, 57, 15));
tcplace.setEnabled(false);
tcplace.setBounds(new Rectangle(239, 136, 94, 20));
btn_choose.setBounds(new Rectangle(63, 202, 81, 23));
btn_choose.setText("选课");
btn_choose.addActionListener(new
                                  Student_choosePanel_btn_choose_actionAdapter(this));
btn_exit.setBounds(new Rectangle(222, 207, 81, 23));
btn_exit.setText("退出");
btn_exit.addActionListener(new
                                Student_choosePanel_btn_exit_actionAdapter(this));
this.add(jLabel1);
this.add(jLabel2);
this.add(jLabel3);
this.add(jcb_course);
this.add(jcb_sno);
this.add(jLabel4);
this.add(tsname);
this.add(jLabel5);
this.add(btn_choose);
this.add(btn_exit);
this.add(jLabel6);
this.add(tcplace);
this.add(tcteacher);
this.add(tcno);
Vector v = sm.selectStudent();          //调用查询所有学生学号方法
```

```
            for (int i = 0; i < v.size(); i++) {
                jcb_sno.addItem(v.get(i).toString()); //将数据库中所有的学号添加到下拉列表中
            }
            Vector v1 = cm.selectCourse();
            for (int i = 0; i < v1.size(); i++) {
            jcb_course.addItem(v1.get(i).toString());  //将数据库中的所有科目添加到下拉列表中
            }
            this.setVisible(true);                      //窗体可见性
            this.setSize(600, 600);                     //窗体大小
        }
        JTextField tsname = new JTextField();
        JComboBox jcb_course = new JComboBox();
        JLabel jLabel1 = new JLabel();
        JLabel jLabel2 = new JLabel();
        JComboBox jcb_sno = new JComboBox();
        JTextField tcno = new JTextField();
        JLabel jLabel3 = new JLabel();
        JLabel jLabel4 = new JLabel();
        JLabel jLabel5 = new JLabel();
        JTextField tcteacher = new JTextField();
        JLabel jLabel6 = new JLabel();
        JTextField tcplace = new JTextField();
        JButton btn_choose = new JButton();
        JButton btn_exit = new JButton();
        StudentManager sm = new StudentManager();       //在程序初始化中定义学生管理对象 sm
        CourseManager cm = new CourseManager();         //在初始化中定义课程管理对象 cm
        public void btn_exit_actionPerformed(ActionEvent e) {
            this.setVisible(false);
        }
    //所选值改变自动执行查找学生方法
    public void jcb_sno_itemStateChanged(ItemEvent e) {
        //调用按学号查询学生信息方法并将学生姓名显示在文本框中
            tsname.setText(sm.findOneStudent(jcb_sno.getSelectedItem().toString()));
        }
        public void jcb_course_itemStateChanged(ItemEvent e) {
        //调用按课程名称查询课程信息方法并将信息显示在相应文本框中
            Course c = cm.findOneCourse(jcb_course.getSelectedItem().toString());
            tcno.setText(c.getCno());
            tcteacher.setText(c.getCteacher());
            tcplace.setText(c.getCplace());
        }
        public void btn_choose_actionPerformed(ActionEvent e) {
            String sno = (String) jcb_sno.getSelectedItem(); //获得下拉列表中选定的学生学号
            String cno = tcno.getText();                    //获得课程名称文本框中的课程代码
            GradeManager cm = new GradeManager();       //定义成绩管理对象 cm
            int i = cm.chooseCourse(sno, cno);              //调用选课方法
            if (i > 0) {
```

```
                JOptionPane.showMessageDialog(this, "选课成功！", "提示信息",
                                              JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this, "添加失败！", "提示信息",
                                              JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    class Student_choosePanel_btn_choose_actionAdapter implements ActionListener {
        private Student_choosePanel adaptee;
        Student_choosePanel_btn_choose_actionAdapter(Student_choosePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_choose_actionPerformed(e);
        }
    }
    class Student_choosePanel_jcb_course_itemAdapter implements ItemListener {
        private Student_choosePanel adaptee;
        Student_choosePanel_jcb_course_itemAdapter(Student_choosePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void itemStateChanged(ItemEvent e) {
            adaptee.jcb_course_itemStateChanged(e);
        }
    }
    class Student_choosePanel_jcb_sno_itemAdapter implements ItemListener {
        private Student_choosePanel adaptee;
        Student_choosePanel_jcb_sno_itemAdapter(Student_choosePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void itemStateChanged(ItemEvent e) {
            adaptee.jcb_sno_itemStateChanged(e);
        }
    }
    class Student_choosePanel_btn_exit_actionAdapter implements ActionListener {
        private Student_choosePanel adaptee;
        Student_choosePanel_btn_exit_actionAdapter(Student_choosePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_exit_actionPerformed(e);
        }
    }
```

6．Add_coursePanel.java

Add_coursePanel 类用于实现添加课程信息界面功能，其界面如图 5-17 所示。

图 5-17　添加课程信息界面

其代码如下：

```
package cvit.com.view;
import cvit.com.manager.CourseManager;
import cvit.com.model.Course;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JLabel;
import java.awt.Rectangle;
import javax.swing.JButton;
import javax.swing.JTextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JOptionPane;
import cvit.com.manager.CourseManager;
//添加课程
public class Add_coursePanel extends JPanel {
    public Add_coursePanel() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
        this.setBackground(new Color(168, 233, 216));
        btn_add.setBounds(new Rectangle(37, 182, 81, 23));
        btn_add.setText("添加");
        btn_add.addActionListener(new Add_coursePanel_btn_add_actionAdapter(this));
        btn_null.setBounds(new Rectangle(153, 182, 81, 23));
        btn_null.setText("清空");
```

```
        btn_null.addActionListener(new Add_coursePanel_btn_null_actionAdapter(this));
        btn_exit.setBounds(new Rectangle(267, 182, 81, 23));
        btn_exit.setText("退出");
        btn_exit.addActionListener(new Add_coursePanel_btn_exit_actionAdapter(this));
        jLabel1.setText("课程类别");
        jLabel1.setBounds(new Rectangle(198, 87, 62, 15));
        jLabel2.setToolTipText("");
        jLabel2.setText("授课教师");
        jLabel2.setBounds(new Rectangle(19, 89, 53, 15));
        tctime.setBounds(new Rectangle(265, 123, 77, 20));
        tcno.setBounds(new Rectangle(93, 45, 77, 20));
        tteacher.setBounds(new Rectangle(91, 85, 77, 20));
        jLabel3.setText("上课时间");
        jLabel3.setBounds(new Rectangle(199, 124, 57, 15));
        jLabel4.setText("课程编码");
        jLabel4.setBounds(new Rectangle(20, 47, 65, 15));
        jLabel5.setText("课程名称");
        jLabel5.setBounds(new Rectangle(201, 47, 60, 15));
        tcplace.setBounds(new Rectangle(93, 123, 77, 20));
        tctype.setBounds(new Rectangle(265, 85, 77, 20));
        tcname.setBounds(new Rectangle(264, 44, 77, 20));
        jLabel6.setText("上课地点");
        jLabel6.setBounds(new Rectangle(19, 124, 67, 15));
        this.add(tteacher);
        this.add(jLabel1);
        this.add(tctime);
        this.add(tcno);
        this.add(jLabel3);
        this.add(jLabel5);
        this.add(tcplace);
        this.add(tctype);
        this.add(tcname);
        this.add(jLabel6);
        this.add(jLabel4);
        this.add(btn_null);
        this.add(btn_add);
        this.add(btn_exit);
        this.add(jLabel2);
        this.setVisible(true);                    //窗体可见性
        this.setSize(600, 600);                   //设置窗体大小
    }
    JButton btn_add = new JButton();
    JButton btn_null = new JButton();
    JButton btn_exit = new JButton();
    JLabel jLabel1 = new JLabel();
    JLabel jLabel2 = new JLabel();
    JTextField tctime = new JTextField();
```

```
JTextField tcno = new JTextField();
JTextField tteacher = new JTextField();
JLabel jLabel3 = new JLabel();
JLabel jLabel4 = new JLabel();
JLabel jLabel5 = new JLabel();
JTextField tcplace = new JTextField();
JTextField tctype = new JTextField();
JTextField tcname = new JTextField();
JLabel jLabel6 = new JLabel();
public void btn_add_actionPerformed(ActionEvent e) {
    String cno = "";                                    //定义空的学生信息属性
    String cname = "";
    String cteacher = "";
    String ctype = "";
    String cplace = "";
    String ctime = "";
    if (!tcno.getText().equals("")) {
        cno = tcno.getText();                           //获得课程名称文本框中的内容
    } else {
        JOptionPane.showMessageDialog(this, "课程编码不能为空！", "提示信息",
                                            JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (!tcname.getText().equals("")) {
        cname = tcname.getText();

    } else {
        JOptionPane.showMessageDialog(this, "课程名称不能为空！", "提示信息",
                                            JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (!tteacher.getText().equals("")) {
        cteacher = tteacher.getText();
    } else {
        JOptionPane.showMessageDialog(this, "授课教师不能为空！", "提示信息",
                                            JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (!tctype.getText().equals("")) {
        ctype = tctype.getText();
    } else {
        JOptionPane.showMessageDialog(this, "课程类别不能为空！", "提示信息",
                                            JOptionPane.ERROR_MESSAGE);
        return;
    }
    if (!tcplace.getText().equals("")) {
        cplace = tcplace.getText();
```

```
        } else {
            JOptionPane.showMessageDialog(this, "上课地点不能为空！", "提示信息",
                                          JOptionPane.ERROR_MESSAGE);
            return;
        }
        if (!tctime.getText().equals("")) {
            ctime = tctime.getText();
        } else {
            JOptionPane.showMessageDialog(this, "上课时间不能为空！", "提示信息",
                                          JOptionPane.ERROR_MESSAGE);
            return;
        }
        Course c = new Course();                    //定义课程信息并对信息封装
        c.setCno(cno);
        c.setCname(cname);
        c.setCteacher(cteacher);
        c.setCtype(ctype);
        c.setCplace(cplace);
        c.setCtime(ctime);
        CourseManager cm = new CourseManager();
        int i = cm.addCourse(c);                    //调用添加课程信息方法
        if (i > 0) {
            JOptionPane.showMessageDialog(this, "添加成功！", "提示信息",
                                          JOptionPane.INFORMATION_MESSAGE);
            tcno.setText("");                       //清空文本框内容
            tcname.setText("");
            tctype.setText("");
            tcplace.setText("");
            tteacher.setText("");
            tctime.setText("");
        } else {
            JOptionPane.showMessageDialog(this, "添加失败！", "提示信息",
                                          JOptionPane.ERROR_MESSAGE);
        }
    }
    public void btn_null_actionPerformed(ActionEvent e) {
        tcno.setText("");                           //清空文本框内容
        tcname.setText("");
        tteacher.setText("");
        tctype.setText("");
        tcplace.setText("");
        tctime.setText("");
    }
    //退出按钮关闭当前页面
    public void btn_exit_actionPerformed(ActionEvent e) {
        this.setVisible(false);
    }
```

```
    }
    class Add_coursePanel_btn_exit_actionAdapter implements ActionListener {
        private Add_coursePanel adaptee;
        Add_coursePanel_btn_exit_actionAdapter(Add_coursePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_exit_actionPerformed(e);
        }
    }
    class Add_coursePanel_btn_null_actionAdapter implements ActionListener {
        private Add_coursePanel adaptee;
        Add_coursePanel_btn_null_actionAdapter(Add_coursePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_null_actionPerformed(e);
        }
    }
    class Add_coursePanel_btn_add_actionAdapter implements ActionListener {
        private Add_coursePanel adaptee;
        Add_coursePanel_btn_add_actionAdapter(Add_coursePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_add_actionPerformed(e);
        }
    }
```

7．Update_coursePanel.java

Update_coursePanel 类用于实现修改课程信息界面功能，其界面如图 5-18 所示。

图 5-18　修改课程信息界面

其代码如下：

```
package cvit.com.view;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JLabel;
import java.awt.Rectangle;
import javax.swing.JButton;
import javax.swing.JTextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import cvit.com.manager.CourseManager.*;
import cvit.com.model.Course.*;
import cvit.com.manager.CourseManager;
import cvit.com.model.Course;
import javax.swing.JOptionPane;
//修改课程信息
public class Update_coursePanel extends JPanel {
    public Update_coursePanel() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
        this.setBackground(new Color(168, 233, 216));
        jLabel1.setText("课程编码");
        jLabel1.setBounds(new Rectangle(20, 47, 65, 15));
        btn_update.setBounds(new Rectangle(153, 182, 81, 23));
        btn_update.setText("修改");
        btn_update.addActionListener(new
                                     Update_coursePanel_btn_update_actionAdapter(this));
        tcno.setBounds(new Rectangle(93, 45, 77, 20));
        btn_select.setBounds(new Rectangle(37, 182, 81, 23));
        btn_select.setText("查找");
        btn_select.addActionListener(new
                                       Update_coursePanel_btn_select_actionAdapter(this));
        jButton3.setBounds(new Rectangle(267, 182, 81, 23));
        jButton3.setText("退出");
        jButton3.addActionListener(new
                                           Update_coursePanel_jButton3_actionAdapter(this));
        jLabel2.setText("课程类别");
        jLabel2.setBounds(new Rectangle(201, 90, 62, 15));
        jLabel3.setText("上课时间");
        jLabel3.setBounds(new Rectangle(200, 127, 57, 15));
```

```
        tcname.setEnabled(false);
        tcname.setBounds(new Rectangle(264, 44, 77, 20));
        jLabel4.setToolTipText("");
        jLabel4.setText("授课教师");
        jLabel4.setBounds(new Rectangle(20, 88, 53, 15));
        tctime.setEnabled(false);
        tctime.setBounds(new Rectangle(265, 123, 77, 20));
        tcteacher.setEnabled(false);
        tcteacher.setBounds(new Rectangle(91, 85, 77, 20));
        jLabel5.setText("上课地点");
        jLabel5.setBounds(new Rectangle(20, 122, 67, 15));
        tcplace.setEnabled(false);
        tcplace.setBounds(new Rectangle(93, 123, 77, 20));
        tctype.setEnabled(false);
        tctype.setBounds(new Rectangle(265, 85, 77, 20));
        jLabel6.setText("课程名称");
        jLabel6.setBounds(new Rectangle(200, 48, 60, 15));
        this.add(jLabel1);
        this.add(btn_update);
        this.add(tcno);
        this.add(btn_select);
        this.add(jButton3);
        this.add(tcname);
        this.add(tctime);
        this.add(tcteacher);
        this.add(tcplace);
        this.add(tctype);
        this.add(jLabel4);
        this.add(jLabel5);
        this.add(jLabel2);
        this.add(jLabel3);
        this.add(jLabel6);
    }
    JLabel jLabel1 = new JLabel();
    JButton btn_update = new JButton();
    JTextField tcno = new JTextField();
    JButton btn_select = new JButton();
    JButton jButton3 = new JButton();
    JLabel jLabel2 = new JLabel();
    JLabel jLabel3 = new JLabel();
    JTextField tcname = new JTextField();
    JLabel jLabel4 = new JLabel();
    JTextField tctime = new JTextField();
    JTextField tcteacher = new JTextField();
    JLabel jLabel5 = new JLabel();
    JTextField tcplace = new JTextField();
    JTextField tctype = new JTextField();
```

```
JLabel jLabel6 = new JLabel();

public void btn_select_actionPerformed(ActionEvent e) {
    String cno = tcno.getText().trim();                    //获得课程编码文本框内容
    CourseManager cm = new CourseManager();
    Course c = cm.findCourse(cno);                         //调用按课程编码查询课程信息方法
    if (c == null) {
        JOptionPane.showMessageDialog(this, "没有该课程！", "提示信息",
                                                JOptionPane.INFORMATION_MESSAGE);
    } else {
        tcno.setEnabled(false);                            //设置文本框是否可用
        tcname.setEnabled(true);
        tcteacher.setEnabled(true);
        tctype.setEnabled(true);
        tcplace.setEnabled(true);
        tctime.setEnabled(true);
        tcno.setText(c.getCno());
        tcname.setText(c.getCname());                      //将查询的结果显示在各文本框中
        tcteacher.setText(c.getCteacher());
        tctype.setText(c.getCtype());
        tcplace.setText(c.getCplace());
        tctime.setText(c.getCtime());
    }
}
public void btn_update_actionPerformed(ActionEvent e) {
    String cno = tcno.getText();                           //获得各文本框内容
```

```
    String cname = tcname.getText();
    String cteacher = tcteacher.getText();
    String ctype = tctype.getText();
    String cplace = tcplace.getText();
    String ctime = tctime.getText();
    CourseManager cm = new CourseManager();    //定义课程管理对象 cm
     //调用修改课程信息方法
    int i = cm.update_Course(cno, cname, cteacher, ctype, cplace, ctime);
    if (i > 0) {
        JOptionPane.showMessageDialog(this, "修改成功！", "提示信息",
                                                JOptionPane.INFORMATION_MESSAGE);
        tcno.setEnabled(true);
        tcname.setEnabled(false);
        tcteacher.setEnabled(false);
        tctype.setEnabled(false);
        tcplace.setEnabled(false);
        tctime.setEnabled(false);
        tcno.setText("");
        tcname.setText("");
```

```
                tctype.setText("");
                tcplace.setText("");
                tcteacher.setText("");
                tctime.setText("");
            } else {
                JOptionPane.showMessageDialog(this, "修改失败！", "提示信息",
                                                    JOptionPane.ERROR_MESSAGE);
            }
        }
        public void jButton3_actionPerformed(ActionEvent e) {
            this.setVisible(false);
        }
    }
    class Update_coursePanel_btn_update_actionAdapter implements ActionListener {
        private Update_coursePanel adaptee;
        Update_coursePanel_btn_update_actionAdapter(Update_coursePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_update_actionPerformed(e);
        }
    }
    class Update_coursePanel_jButton3_actionAdapter implements ActionListener {
        private Update_coursePanel adaptee;
        Update_coursePanel_jButton3_actionAdapter(Update_coursePanel adaptee) {
            this.adaptee = adaptee;
```

```
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.jButton3_actionPerformed(e);
        }
    }
    class Update_coursePanel_btn_select_actionAdapter implements ActionListener {
        private Update_coursePanel adaptee;
        Update_coursePanel_btn_select_actionAdapter(Update_coursePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_select_actionPerformed(e);
        }
    }
```

8．Delete_coursePanel.java

Delete_coursePanel.java 类用于实现删除课程信息界面功能，其界面如图 5-19 所示。

图 5-19　删除课程信息界面

其代码如下：

```
package cvit.com.view;
import cvit.com.manager.CourseManager;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JTextField;
import java.awt.Rectangle;
import javax.swing.JLabel;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JOptionPane;
//删除科目
public class Delete_coursePanel extends JPanel {
    public Delete_coursePanel() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
        this.setBackground(new Color(168, 233, 216));
        tcno.setBounds(new Rectangle(78, 76, 130, 20));
        jLabel1.setText("课程编号");
        jLabel1.setBounds(new Rectangle(8, 81, 66, 15));
        btn_delete.setBounds(new Rectangle(228, 72, 71, 23));
        btn_delete.setText("删除");
        btn_delete.addActionListener(new
                                     Delete_coursePanel_btn_delete_actionAdapter(this));
        btn_exit.setBounds(new Rectangle(320, 72, 69, 23));
```

```
        btn_exit.setText("退出");
        btn_exit.addActionListener(new
                                   Delete_coursePanel_btn_exit_actionAdapter(this));
        this.add(tcno);
        this.add(jLabel1);
        this.add(btn_delete);
        this.add(btn_exit);
        this.setVisible(true);                    //可见性
        this.setSize(600, 600);                   //窗体大小
    }
    JTextField tcno = new JTextField();
    JLabel jLabel1 = new JLabel();
    JButton btn_delete = new JButton();
    JButton btn_exit = new JButton();
    public void btn_delete_actionPerformed(ActionEvent e) {
        String cno = tcno.getText();              //获得课程编码
        CourseManager cm = new CourseManager();
        int ok = JOptionPane.showConfirmDialog(this,
                                               "是否删除" + tcno.getText() + "课程？","提示
                                               信息",JOptionPane.OK_CANCEL_OPTION);
        if (ok == JOptionPane.OK_OPTION) {
            int i = cm.deleteCourse(cno);         //调用删除课程信息方法
            if (i > 0) {
                JOptionPane.showMessageDialog(this, "删除成功！", "提示信息",
                                              JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this, "你要删除的课程不存在！",
                                              "提示信息",JOptionPane.ERROR_MESSAGE);
                tcno.setText("");
            }
            tcno.setText("");
        }
    }
    public void btn_exit_actionPerformed(ActionEvent e) {
        this.setVisible(false);
    }
    class Delete_coursePanel_btn_delete_actionAdapter implements ActionListener {
        private Delete_coursePanel adaptee;
        Delete_coursePanel_btn_delete_actionAdapter(Delete_coursePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_delete_actionPerformed(e);
        }
    }
}
class Delete_coursePanel_btn_exit_actionAdapter implements ActionListener {
```

```
        private Delete_coursePanel adaptee;
        Delete_coursePanel_btn_exit_actionAdapter(Delete_coursePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_exit_actionPerformed(e);
        }
    }
```

9．Add_gradePanel.java

Add_gradePanel 类用于实现成绩添加界面功能，其界面如图 5-20 所示。

图 5-20　成绩添加界面

其代码如下：

```
package cvit.com.view;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JLabel;
import java.awt.Rectangle;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JTextField;
import cvit.com.manager.CourseManager;
import cvit.com.manager.StudentManager;
import java.util.Vector;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import cvit.com.model.Course;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import cvit.com.model.Grade;
import javax.swing.JOptionPane;
import cvit.com.manager.GradeManager;
//添加分数
public class Add_gradePanel extends JPanel {
    public Add_gradePanel() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
        this.setBackground(new Color(168, 233, 216));
        jLabel1.setText("学号");
        jLabel1.setBounds(new Rectangle(30, 22, 42, 15));
        jLabel2.setText("姓名");
        jLabel2.setBounds(new Rectangle(200, 20, 42, 15));
        jLabel3.setText("课程名称");
        jLabel3.setBounds(new Rectangle(23, 76, 55, 15));
        jLabel4.setText("课程编码");
        jLabel4.setBounds(new Rectangle(201, 74, 57, 15));
        jLabel5.setText("授课教师");
        jLabel5.setBounds(new Rectangle(15, 129, 61, 15));
        jLabel6.setText("成绩");
        jLabel6.setBounds(new Rectangle(202, 129, 42, 15));
        btn_add.setBounds(new Rectangle(76, 176, 81, 23));
        btn_add.setText("增加");
        btn_add.addActionListener(new addcjPanel1_btn_add_actionAdapter(this));
        btn_exit.setBounds(new Rectangle(233, 175, 81, 23));
        btn_exit.setText("退出");
        btn_exit.addActionListener(new addcjPanel1_btn_exit_actionAdapter(this));
        jcb_sno.setBounds(new Rectangle(91, 17, 89, 23));
        jcb_sno.addItemListener(new addcjPanel1_jcb_sno_itemAdapter(this));
        jcb_cname.setBounds(new Rectangle(92, 72, 89, 23));
        jcb_cname.addItemListener(new addcjPanel1_jcb_cname_itemAdapter(this));
        tsname.setEnabled(false);
        tsname.setBounds(new Rectangle(261, 19, 77, 20));
        tcno.setEnabled(false);
        tcno.setBounds(new Rectangle(260, 74, 77, 20));
        tcteacher.setEnabled(false);
        tcteacher.setBounds(new Rectangle(91, 127, 77, 20));
        tgrade.setBounds(new Rectangle(262, 127, 77, 20));
        this.add(jLabel2);
        this.add(jcb_sno);
        this.add(tsname);
```

```
            this.add(jLabel1);
            this.add(tcno);
            this.add(jLabel4);
            this.add(jcb_cname);
            this.add(jLabel3);
            this.add(tcteacher);
            this.add(jLabel5);
            this.add(jLabel6);
            this.add(tgrade);
            this.add(btn_exit);
            this.add(btn_add);
            this.setVisible(true);
            this.setSize(600, 600);
            Vector v = sm.selectStudent();              //调用查询所有学号方法
            for (int i = 0; i < v.size(); i++) {
                jcb_sno.addItem(v.get(i).toString()); //将数据库所有学号添加到下拉列表中
            }
            Vector v1 = cm.selectCourse();              //调用查询所有科目名称方法
            for (int i = 0; i < v1.size(); i++) {
              jcb_cname.addItem(v1.get(i).toString()); //将数据库中所有的课程名称添加到列表中
            }
        }
        JLabel jLabel1 = new JLabel();
        JLabel jLabel2 = new JLabel();
        JLabel jLabel3 = new JLabel();
        JLabel jLabel4 = new JLabel();
        JLabel jLabel5 = new JLabel();
        JLabel jLabel6 = new JLabel();
        JButton btn_add = new JButton();
        JButton btn_exit = new JButton();
        JComboBox jcb_sno = new JComboBox();
        JComboBox jcb_cname = new JComboBox();
        JTextField tsname = new JTextField();
        JTextField tcno = new JTextField();
        JTextField tcteacher = new JTextField();
        JTextField tgrade = new JTextField();
        StudentManager sm = new StudentManager();  //初始化中定义学生管理对象
        CourseManager cm = new CourseManager();   //初始化中定义课程管理对象
        public static void main(String[] args) {Add_gradePanel sm = new Add_gradePanel();
    }
        public void jcb_sno_itemStateChanged(ItemEvent e) {
            //调用按学号查询学生信息方法
            tsname.setText(sm.findOneStudent(jcb_sno.getSelectedItem().toString()));
        }
        public void jcb_cname_itemStateChanged(ItemEvent e) {
            //调用按课程名称查询课程信息方法
            Course c = cm.findOneCourse(jcb_cname.getSelectedItem().toString());
```

```
            tcno.setText(c.getCno());
            tcteacher.setText(c.getCteacher());
            }
        public void btn_exit_actionPerformed(ActionEvent e) {
            this.setVisible(false);
        }
        public void btn_add_actionPerformed(ActionEvent e) {
            String sno = (String) jcb_sno.getSelectedItem();
            String cno = tcno.getText();
            String grade = tgrade.getText();
            Grade g = new Grade();
            g.setSno(sno);
            g.setCno(cno);
            g.setGrade(grade);
            GradeManager gm = new GradeManager();
            int i = gm.addGrade(g);                    //调用添加成绩方法
            if (i > 0) {
                JOptionPane.showMessageDialog(this, "添加成功！", "提示信息",
                                                    JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this, "学生与实际选课科目不符，请核对！",
                            "提示信息",JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    class addcjPanel1_btn_add_actionAdapter implements ActionListener {
        private Add_gradePanel adaptee;
        addcjPanel1_btn_add_actionAdapter(Add_gradePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_add_actionPerformed(e);
        }
    }
    class addcjPanel1_btn_exit_actionAdapter implements ActionListener {
        private Add_gradePanel adaptee;
        addcjPanel1_btn_exit_actionAdapter(Add_gradePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_exit_actionPerformed(e);
        }
    }
    class addcjPanel1_jcb_cname_itemAdapter implements ItemListener {
        private Add_gradePanel adaptee;
        addcjPanel1_jcb_cname_itemAdapter(Add_gradePanel adaptee) {
            this.adaptee = adaptee;
```

```
        }
        public void itemStateChanged(ItemEvent e) {
            adaptee.jcb_cname_itemStateChanged(e);
        }
    }
    class addcjPanel1_jcb_sno_itemAdapter implements ItemListener {
        private Add_gradePanel adaptee;
        addcjPanel1_jcb_sno_itemAdapter(Add_gradePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void itemStateChanged(ItemEvent e) {
            adaptee.jcb_sno_itemStateChanged(e);
        }
    }
```

10．Update_gradePanel.java

Update_gradePanel 类用于实现修改成绩界面功能，其界面如图 5-21 所示。

图 5-21　修改成绩界面

其代码如下：

```
package cvit.com.view;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JTextField;
import java.awt.Rectangle;
import javax.swing.JButton;
import javax.swing.JLabel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JOptionPane;
import cvit.com.manager.GradeManager;
import cvit.com.model.Grade;
import javax.swing.JComboBox;
import java.util.Vector;
import cvit.com.manager.StudentManager;
```

```
import cvit.com.manager.CourseManager;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import cvit.com.model.Course;
//修改分数
public class Update_gradePanel extends JPanel {
    public Update_gradePanel() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
        this.setBackground(new Color(168, 233, 216));
        btn_update.setBounds(new Rectangle(145, 238, 78, 23));
        btn_update.setText("修改");
        btn_update.addActionListener(new
                Update_gradePanel_btn_update_actionAdapter(this));
        btn_exit.setBounds(new Rectangle(232, 239, 71, 23));
        btn_exit.setText("退出");
        btn_exit.addActionListener(new Update_gradePanel_btn_exit_actionAdapter(this));
        jLabel1.setText("学号");
        jLabel1.setBounds(new Rectangle(62, 45, 42, 15));
        jLabel2.setText("成绩");
        jLabel2.setBounds(new Rectangle(62, 193, 58, 15));
        t2.setEnabled(false);
        t2.setBounds(new Rectangle(136, 144, 77, 20));
        jLabel3.setText("课程编号");
        jLabel3.setBounds(new Rectangle(55, 146, 62, 15));
        t3.setEnabled(false);
        t3.setBounds(new Rectangle(144, 193, 77, 20));
        btn_select.setBounds(new Rectangle(55, 237, 81, 23));
        btn_select.setText("查询");
        btn_select.addActionListener(new
                Update_gradePanel_btn_select_actionAdapter(this));
        jcb_sno.setBounds(new Rectangle(135, 43, 98, 23));
        jcb_sno.addItemListener(new Update_gradePanel_jcb_sno_itemAdapter(this));
        jcb_cname.setBounds(new Rectangle(137, 96, 101, 23));
        jcb_cname.addItemListener(new Update_gradePanel_jcb_cname_itemAdapter(this));
        jLabel4.setText("课程名称");
        jLabel4.setBounds(new Rectangle(60, 99, 56, 15));
        this.add(btn_update);
        this.add(btn_select);
        this.add(btn_exit);
        this.add(t3);
```

```
        this.add(jLabel2);
        this.add(jLabel3);
        this.add(t2);
        this.add(jcb_cname);
        this.add(jcb_sno);
        this.add(jLabel1);
        this.add(jLabel4);
        Vector v = sm.selectStudent();                    //获得所有学号
        for (int i = 0; i < v.size(); i++) {
            jcb_sno.addItem(v.get(i).toString());         //将学号添加到下拉列表中
        }
        Vector v1 = cm.selectCourse();                    //获得所有课程名称
        for (int i = 0; i < v1.size(); i++) {
            jcb_cname.addItem(v1.get(i).toString());      //将所有课程名称添加到下拉列表中
        }
        this.setVisible(true);
        this.setSize(600, 600);
    }
    JButton btn_update = new JButton();
    JButton btn_exit = new JButton();
    JLabel jLabel1 = new JLabel();
    JLabel jLabel2 = new JLabel();
    JTextField t2 = new JTextField();
    JLabel jLabel3 = new JLabel();
    JTextField t3 = new JTextField();
    JButton btn_select = new JButton();
    JComboBox jcb_sno = new JComboBox();
    StudentManager sm = new StudentManager();
    CourseManager cm = new CourseManager();
    JComboBox jcb_cname = new JComboBox();
    JLabel jLabel4 = new JLabel();
    public void btn_exit_actionPerformed(ActionEvent e) {
        this.setVisible(false);
    }
    public void btn_select_actionPerformed(ActionEvent e) {
        String sno = jcb_sno.getSelectedItem().toString();
        String cno = t2.getText();
        GradeManager gm = new GradeManager();
        Grade g = gm.findGrade(sno, cno);                 //调用按学号和课程编号查询选课方法
        if (g == null) {
            JOptionPane.showMessageDialog(this, "与学生实际选课不符，请核对！",
                                "提示信息",JOptionPane.INFORMATION_MESSAGE);
        } else {
            t2.setEnabled(false);
            t3.setEnabled(true);
            t2.setText(g.getCno());
            t3.setText(g.getGrade());
```

```
            }
        }
        public void btn_update_actionPerformed(ActionEvent e) {
            String sno = jcb_sno.getSelectedItem().toString();
            String cno = t2.getText();
            String grade = t3.getText();
            GradeManager gm = new GradeManager();
            int i = gm.updateGrade(sno, cno, grade);                    //调用修改成绩方法
            if (i > 0) {
                JOptionPane.showMessageDialog(this, "修改成功！", "提示信息",
                                                    JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this, "修改失败！", "提示信息",
                                                    JOptionPane.ERROR_MESSAGE);
            }
        }
        public void jcb_cname_itemStateChanged(ItemEvent e) {
        //调用按课程名称查询课程信息方法
            Course c = cm.findOneCourse(jcb_cname.getSelectedItem().toString());
            t2.setText(c.getCno());
        }
        public void jcb_sno_itemStateChanged(ItemEvent e) {
        }
    }
    class Update_gradePanel_jcb_sno_itemAdapter implements ItemListener {
        private Update_gradePanel adaptee;
        Update_gradePanel_jcb_sno_itemAdapter(Update_gradePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void itemStateChanged(ItemEvent e) {
            adaptee.jcb_sno_itemStateChanged(e);
        }
    }
    class Update_gradePanel_jcb_cname_itemAdapter implements ItemListener {
        private Update_gradePanel adaptee;
        Update_gradePanel_jcb_cname_itemAdapter(Update_gradePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void itemStateChanged(ItemEvent e) {
            adaptee.jcb_cname_itemStateChanged(e);
        }
    }
    class Update_gradePanel_btn_update_actionAdapter implements ActionListener {
        private Update_gradePanel adaptee;
        Update_gradePanel_btn_update_actionAdapter(Update_gradePanel adaptee) {
            this.adaptee = adaptee;
        }
```

```
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_update_actionPerformed(e);
        }
    }
    class Update_gradePanel_btn_select_actionAdapter implements ActionListener {
        private Update_gradePanel adaptee;
        Update_gradePanel_btn_select_actionAdapter(Update_gradePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_select_actionPerformed(e);
        }
    }
    class Update_gradePanel_btn_exit_actionAdapter implements ActionListener {
        private Update_gradePanel adaptee;
        Update_gradePanel_btn_exit_actionAdapter(Update_gradePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_exit_actionPerformed(e);
        }
    }
```

11. Select_studentnoPanel1.java

Select_studentnoPanel1 类用于实现按学号查询学生信息界面功能，其界面如图 5-22 所示。

图 5-22　按学号查询学生信息界面

其代码如下：

```
    package cvit.com.view;
    import javax.swing.JPanel;
    import java.awt.Color;
    import javax.swing.JLabel;
```

```
import java.awt.Rectangle;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JOptionPane;
import cvit.com.model.Student;
import cvit.com.manager.StudentManager;
//按学号查询
public class Select_studentnoPanel1 extends JPanel {
    public Select_studentnoPanel1() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
        this.setBackground(new Color(168, 233, 216));
        jLabel1.setText("籍贯");
        jLabel1.setBounds(new Rectangle(16, 197, 42, 15));
        jLabel2.setText("姓名");
        jLabel2.setBounds(new Rectangle(187, 35, 42, 15));
        tsno.setBounds(new Rectangle(79, 33, 77, 20));
        tscollege.setEnabled(false);
        tscollege.setBounds(new Rectangle(79, 153, 77, 20));
        jLabel3.setText("专业");
        jLabel3.setBounds(new Rectangle(188, 154, 42, 15));
        jLabel4.setText("学号");
        jLabel4.setBounds(new Rectangle(15, 36, 42, 15));
        jLabel5.setText("入学时间");
        jLabel5.setBounds(new Rectangle(175, 112, 67, 15));
        tsname.setEnabled(false);
        tsname.setBounds(new Rectangle(250, 32, 77, 20));
        tsmajor.setEnabled(false);
        tsmajor.setBounds(new Rectangle(251, 156, 77, 20));
        jLabel6.setText("民族");
        jLabel6.setBounds(new Rectangle(184, 75, 42, 15));
        tstime.setEnabled(false);
        tstime.setBounds(new Rectangle(251, 111, 77, 20));
        tssex.setEnabled(false);
        tssex.setToolTipText("");
        tssex.setBounds(new Rectangle(77, 73, 77, 20));
        jLabel7.setText("性别");
        jLabel7.setBounds(new Rectangle(14, 74, 42, 15));
        tsbirth.setEnabled(false);
```

```
        tsbirth.setBounds(new Rectangle(79, 111, 77, 20));
        jLabel8.setText("出生日期");
        jLabel8.setBounds(new Rectangle(12, 112, 53, 15));
        tsethnix.setEnabled(false);
        tsethnix.setBounds(new Rectangle(251, 73, 77, 20));
        tshome.setEnabled(false);
        tshome.setBounds(new Rectangle(77, 198, 251, 20));
        jLabel9.setText("学院");
        jLabel9.setBounds(new Rectangle(15, 154, 42, 15));
        btn_exit.setBounds(new Rectangle(245, 247, 81, 23));
        btn_exit.setText("退出");
        btn_exit.addActionListener(new
                                   Select_studentnoPanel1_btn_exit_actionAdapter(this));
        btn_select.setBounds(new Rectangle(44, 245, 81, 23));
        btn_select.setText("查询");
        btn_select.addActionListener(new
                                     Select_studentnoPanel1_btn_select_actionAdapter(this));
        this.add(jLabel1);
        this.add(jLabel2);
        this.add(tsno);
        this.add(tscollege);
        this.add(jLabel3);
        this.add(jLabel4);
        this.add(jLabel5);
        this.add(tsname);
        this.add(tsmajor);
        this.add(jLabel6);
        this.add(tstime);
        this.add(tssex);
        this.add(jLabel7);
        this.add(tsbirth);
        this.add(jLabel8);
        this.add(tsethnix);
        this.add(tshome);
        this.add(jLabel9);
        this.add(btn_select);
        this.add(btn_exit);
    }
    JLabel jLabel1 = new JLabel();
    JLabel jLabel2 = new JLabel();
    JTextField tsno = new JTextField();
    JTextField tscollege = new JTextField();
    JLabel jLabel3 = new JLabel();
    JLabel jLabel4 = new JLabel();
    JLabel jLabel5 = new JLabel();
    JTextField tsname = new JTextField();
    JTextField tsmajor = new JTextField();
```

```
        JLabel jLabel6 = new JLabel();
        JTextField tstime = new JTextField();
        JTextField tssex = new JTextField();
        JLabel jLabel7 = new JLabel();
        JTextField tsbirth = new JTextField();
        JLabel jLabel8 = new JLabel();
        JTextField tsethnix = new JTextField();
        JTextField tshome = new JTextField();
        JLabel jLabel9 = new JLabel();
        JButton btn_exit = new JButton();
        JButton btn_select = new JButton();
        public void btn_select_actionPerformed(ActionEvent e) {
            String sno = tsno.getText().trim();
            StudentManager sm = new StudentManager();
            Student s = sm.findStudent(sno);                    //调用按学号查询学生信息方法
            if (s == null) {
                JOptionPane.showMessageDialog(this, "没有该名同学！", "提示信息",
                                              JOptionPane.INFORMATION_MESSAGE);
                tsno.setText("");
            } else {
                tsno.setText(s.getSno());                       //将查询到的信息显示在各文本框中
                tsname.setText(s.getSname());
                tssex.setText(s.getSsex());
                tsethnix.setText(s.getSethnix());
                tsbirth.setText(s.getSbirth());
                tstime.setText(s.getSyear());
                tscollege.setText(s.getScollege());
                tsmajor.setText(s.getSmajor());
                tshome.setText(s.getShome());
            }
        }
        public void btn_exit_actionPerformed(ActionEvent e) {
            this.setVisible(false);
        }
    }
    class Select_studentnoPanel1_btn_exit_actionAdapter implements ActionListener {
        private Select_studentnoPanel1 adaptee;
        Select_studentnoPanel1_btn_exit_actionAdapter(Select_studentnoPanel1 adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_exit_actionPerformed(e);
        }
    }
    class Select_studentnoPanel1_btn_select_actionAdapter implements ActionListener {
        private Select_studentnoPanel1 adaptee;
        Select_studentnoPanel1_btn_select_actionAdapter(Select_studentnoPanel1adaptee) {
```

```
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_select_actionPerformed(e);
        }
    }
```

12．Select_studentnamePanel.java

Select_studentnamePanel 类用于实现按姓名查询学生信息界面功能，其界面如图 5-23 所示。

图 5-23　按姓名查询学生信息界面

其代码如下：

```
package cvit.com.view;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JButton;
import java.awt.Rectangle;
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JList;
import cvit.com.manager.StudentManager;
import java.util.Vector;
//按学生名查询
public class Select_studentnamePanel extends JPanel {
    public Select_studentnamePanel() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
```

```
            this.setLayout(null);
            this.setBackground(new Color(168, 233, 216));
            jButton1.setBounds(new Rectangle(314, 36, 81, 23));
            jButton1.setText("退出");
            jButton1.addActionListener(new
                                    Select_studentnamePanel_jButton1_actionAdapter(this));
            jLabel1.setText("学生姓名");
            jLabel1.setBounds(new Rectangle(25, 38, 71, 15));
            jButton2.setBounds(new Rectangle(246, 38, 64, 23));
            jButton2.setText("查询");
            jButton2.addActionListener(new
                                    Select_studentnamePanel_jButton2_actionAdapter(this));
            tsname.setBounds(new Rectangle(108, 38, 126, 20));
            list.setBounds(new Rectangle(20, 84, 442, 70));
            this.add(jButton1);
            this.add(jLabel1);
            this.add(jButton2);
            this.add(tsname);
            this.add(list);
            this.setVisible(true);
            this.setSize(600, 600);
        }
        JButton jButton1 = new JButton();
        JLabel jLabel1 = new JLabel();
        JButton jButton2 = new JButton();
        JTextField tsname = new JTextField();
        JList list = new JList();
        public void jButton1_actionPerformed(ActionEvent e) {
            this.setVisible(false);
        }
        public void jButton2_actionPerformed(ActionEvent e) {
            String sname = tsname.getText().trim();
            StudentManager sm = new StudentManager();
            Vector v = sm.findStudent_name(sname); //调用按学生姓名查询学生信息方法
            list.setListData(v);                    //将 Vector 集合中所有的信息添加到 Jlist 控件中
        }
    }
    class Select_studentnamePanel_jButton2_actionAdapter implements ActionListener {
        private Select_studentnamePanel adaptee;
        Select_studentnamePanel_jButton2_actionAdapter(Select_studentnamePanel
                adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.jButton2_actionPerformed(e);
        }
    }
```

```
class Select_studentnamePanel_jButton1_actionAdapter implements ActionListener {
    private Select_studentnamePanel adaptee;
    Select_studentnamePanel_jButton1_actionAdapter(Select_studentnamePanel
            adaptee) {
        this.adaptee = adaptee;
    }
    public void actionPerformed(ActionEvent e) {
        adaptee.jButton1_actionPerformed(e);
    }
}
```

13．Select_studentsexPanel.java

Select_studentsexPanel 类用于实现按性别查询学生信息界面功能，其界面如图 5-24 所示。

图 5-24　按性别查询学生信息界面

其代码如下：

```
package cvit.com.view;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JButton;
import java.awt.Rectangle;
import javax.swing.JLabel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JList;
import cvit.com.manager.StudentManager;
import java.util.Vector;
import javax.swing.JComboBox;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
//按学生性别查询
public class Select_studentsexPanel extends JPanel {
    public Select_studentsexPanel() {
```

```
            try {
                jbInit();
            } catch (Exception exception) {
                exception.printStackTrace();
            }
        }
        private void jbInit() throws Exception {
            this.setLayout(null);
            this.setBackground(new Color(168, 233, 216));
            jButton1.setBounds(new Rectangle(257, 40, 81, 23));
            jButton1.setText("退出");
            jButton1.addActionListener(new
                                       Select_studentsexPanel_jButton1_actionAdapter(this));
            jLabel1.setText("性别");
            jLabel1.setBounds(new Rectangle(84, 46, 49, 15));
            list.setBounds(new Rectangle(37, 104, 421, 112));
            jcb_sex.setBounds(new Rectangle(155, 40, 63, 23));
            jcb_sex.addItemListener(new Select_studentsexPanel_jcb_sex_itemAdapter(this));
            this.add(list);
            this.add(jLabel1);
            this.add(jcb_sex);
            this.add(jButton1);
            jcb_sex.addItem("男");
            jcb_sex.addItem("女");
        }
        JButton jButton1 = new JButton();
        JLabel jLabel1 = new JLabel();
        JList list = new JList();
        JComboBox jcb_sex = new JComboBox();
        public void jButton1_actionPerformed(ActionEvent e) {
            this.setVisible(false);
        }
            public void jcb_sex_itemStateChanged(ItemEvent e) {
            String ssex = jcb_sex.getSelectedItem().toString();
            StudentManager sm = new StudentManager();
            Vector v = sm.findStudentsex1(ssex);                    //调用按性别查询学生信息方法
            list.setListData(v);                                    //将查询到的信息显示在 Jlist 控件中
        }
    }
    class Select_studentsexPanel_jcb_sex_itemAdapter implements ItemListener {
        private Select_studentsexPanel adaptee;
        Select_studentsexPanel_jcb_sex_itemAdapter(Select_studentsexPanel adaptee) {
            this.adaptee = adaptee;
        }

        public void itemStateChanged(ItemEvent e) {
            adaptee.jcb_sex_itemStateChanged(e);
```

```
        }
    }
    class Select_studentsexPanel_jButton1_actionAdapter implements ActionListener {
        private Select_studentsexPanel adaptee;
        Select_studentsexPanel_jButton1_actionAdapter(Select_studentsexPanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.jButton1_actionPerformed(e);
        }
    }
```

14．Select_studentmajorPanel1.java

Select_studentmajorPanel1 类用于实现按专业查询学生信息界面功能，其界面如图 5-25 所示。

图 5-25　按专业查询学生信息界面

其代码如下：

```
package cvit.com.view;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JButton;
import java.awt.Rectangle;
import javax.swing.JLabel;
import javax.swing.JList;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import cvit.com.manager.StudentManager;
import java.util.Vector;
import javax.swing.JComboBox;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
//按专业查询
public class Select_studentmajorPanel1 extends JPanel {
```

```
    public Select_studentmajorPanel1() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
        this.setBackground(new Color(168, 233, 216));
        btn_exit.setBounds(new Rectangle(314, 36, 81, 23));
        btn_exit.setText("退出");
        btn_exit.addActionListener(new
                                   Select_srudentmajorPanel1_btn_exit_actionAdapter(this));
        jLabel1.setText("所学专业");
        jLabel1.setBounds(new Rectangle(34, 38, 62, 15));
        list1.setBounds(new Rectangle(31, 76, 428, 125));
        jcb_major.setBounds(new Rectangle(119, 31, 100, 23));
        jcb_major.addItemListener(new
                                  Select_studentmajorPanel1_jcb_major_itemAdapter(this));
        this.add(btn_exit);
        this.add(jLabel1);
        this.add(list1);
        this.add(jcb_major);
        Vector v = sm.selectStudentmajor();              //调用查询所有专业方法
        for (int i = 0; i < v.size(); i++) {
            jcb_major.addItem(v.get(i).toString());      //将查询的所有专业添加到下拉列表中
        }
```

```
        this.setVisible(true);
        this.setSize(600, 600);
    }
    JButton btn_exit = new JButton();
    JLabel jLabel1 = new JLabel();
    JList list1 = new JList();
    StudentManager sm = new StudentManager();
    JComboBox jcb_major = new JComboBox();
    public void btn_exit_actionPerformed(ActionEvent e) {
        this.setVisible(false);
    }
    public void jcb_major_itemStateChanged(ItemEvent e) {
        String smajor = jcb_major.getSelectedItem().toString();
        StudentManager sm = new StudentManager();
        Vector v1 = sm.findStudentmajor(smajor);       //调用根据专业查询学生信息方法
        list1.setListData(v1);
    }
}
```

```
class Select_studentmajorPanel1_jcb_major_itemAdapter implements ItemListener {
    private Select_studentmajorPanel1 adaptee;
    Select_studentmajorPanel1_jcb_major_itemAdapter(Select_studentmajorPanel1
            adaptee) {
        this.adaptee = adaptee;
    }
    public void itemStateChanged(ItemEvent e) {
        adaptee.jcb_major_itemStateChanged(e);
    }
}
class Select_srudentmajorPanel1_btn_exit_actionAdapter implements
        ActionListener {
    private Select_studentmajorPanel1 adaptee;
    Select_srudentmajorPanel1_btn_exit_actionAdapter(Select_studentmajorPanel1 adaptee) {
        this.adaptee = adaptee;
    }
    public void actionPerformed(ActionEvent e) {
        adaptee.btn_exit_actionPerformed(e);
    }
}
```

15．Select_studentcollegePanel.java

Select_studentcollegePanel 类用于实现按学院查询学生信息界面功能，其界面如图 5-26 所示。

图 5-26　按学院查询学生信息界面

其代码如下：

```
package cvit.com.view;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JLabel;
import java.awt.Rectangle;
import javax.swing.JButton;
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import cvit.com.manager.StudentManager;
import javax.swing.JList;
import java.util.Vector;
import javax.swing.JComboBox;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import javax.swing.JScrollPane;
//按分院查询
pub lic class Select_studentcollegePanel extends JPanel {
    public Select_studentcollegePanel() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
        this.setBackground(new Color(168, 233, 216));
        jLabel1.setText("所在分院");
        jLabel1.setBounds(new Rectangle(51, 38, 62, 15));
        list1.setBounds(new Rectangle(16, 78, 446, 172));
        btn_exit.setBounds(new Rectangle(282, 31, 81, 23));
        btn_exit.setText("退出");
        btn_exit.addActionListener(new
                                   Select_studentcollegePanel_btn_exit_actionAdapter(this));
        jcb_college.setBounds(new Rectangle(136, 34, 102, 23));
        jcb_college.addItemListener(new
                                    Select_studentcollegePanel_jcb_college_itemAdapter(this));
        this.add(list1);
        this.add(jcb_college);
        this.add(jLabel1);
        this.add(btn_exit);
        this.setVisible(true);
        this.setSize(600, 600);
        Vector v = sm.selectStudentcollege();          //调用查询所有分院方法
        for (int i = 0; i < v.size(); i++) {
            jcb_college.addItem(v.get(i).toString());  //将查询结果显示在下拉列表中
        }
    }
    JLabel jLabel1 = new JLabel();
    Vector v1 = null;
    JList list1 = new JList();
    JButton btn_exit = new JButton();
    JComboBox jcb_college = new JComboBox();
```

```
        StudentManager sm = new StudentManager();
        public void btn_actionPerformed(ActionEvent e) {
            String scollege = jcb_college.getSelectedItem().toString();
            StudentManager sm = new StudentManager();
            Vector v1 = sm.findStudentcollege(scollege);

            list1.setListData(v1);
        }
        public void btn_exit_actionPerformed(ActionEvent e) {
            this.setVisible(false);
        }
        public void jcb_college_itemStateChanged(ItemEvent e) {
            String scollege = jcb_college.getSelectedItem().toString();
            StudentManager sm = new StudentManager();
            Vector v1 = sm.findStudentcollege(scollege);    //调用按分院查询学生信息方法
            list1.setListData(v1);                          //将查到的学生信息显示在 Jlist 控件中
        }
    }
    class Select_studentcollegePanel_jcb_college_itemAdapter implements
            ItemListener {
        private Select_studentcollegePanel adaptee;
        Select_studentcollegePanel_jcb_college_itemAdapter(
                Select_studentcollegePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void itemStateChanged(ItemEvent e) {
            adaptee.jcb_college_itemStateChanged(e);
        }
    }
    class Select_studentcollegePanel_btn_exit_actionAdapter implements
            ActionListener {
        private Select_studentcollegePanel adaptee;
        Select_studentcollegePanel_btn_exit_actionAdapter(
                Select_studentcollegePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_exit_actionPerformed(e);
        }
    }
```

16．Select_coursecnamePanel.java

Select_coursecnamePanel 类用于实现按课程名称查询课程信息界面功能，其界面如图 5-27 所示。

图 5-27　按课程名称查询课程信息界面

其代码如下：

```
package cvit.com.view;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JLabel;
import java.awt.Rectangle;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import cvit.com.manager.CourseManager;
import cvit.com.model.Course;
import javax.swing.JComboBox;
import java.util.Vector;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
//根据课程名称查询
public class Select_coursecnamePanel extends JPanel {
    public Select_coursecnamePanel() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
        this.setBackground(new Color(168, 233, 216));
        jLabel1.setToolTipText("");
        jLabel1.setText("授课教师");
        jLabel1.setBounds(new Rectangle(18, 86, 53, 15));
        jLabel2.setText("课程类别");
```

```
        jLabel2.setBounds(new Rectangle(198, 87, 62, 15));
        tctime.setEnabled(false);
        tctime.setBounds(new Rectangle(265, 123, 77, 20));
        tcno.setEnabled(false);
        tcno.setBounds(new Rectangle(266, 40, 77, 20));
        tcteacher.setEnabled(false);
        tcteacher.setBounds(new Rectangle(91, 85, 93, 20));
        jLabel3.setText("课程名称");
        jLabel3.setBounds(new Rectangle(20, 45, 60, 15));
        jLabel4.setText("课程编码");
        jLabel4.setBounds(new Rectangle(193, 42, 65, 15));
        jLabel5.setText("上课时间");
        jLabel5.setBounds(new Rectangle(199, 124, 57, 15));
        tctype.setEnabled(false);
        tctype.setBounds(new Rectangle(265, 85, 77, 20));
        tcplace.setEnabled(false);
        tcplace.setBounds(new Rectangle(93, 123, 90, 20));
        jLabel6.setText("上课地点");
        jLabel6.setBounds(new Rectangle(19, 124, 67, 15));
        btn_exit.setBounds(new Rectangle(22, 172, 81, 23));
        btn_exit.setText("退出");
        btn_exit.addActionListener(new
                                   Select_coursePanel_btn_exit_actionAdapter(this));
        jcb_cname.setBounds(new Rectangle(89, 41, 97, 23));
        jcb_cname.addItemListener(new
                                   Select_coursecnamePanel_jcb_cname_itemAdapter(this));
        this.add(jLabel2);
        this.add(tctime);
        this.add(tcteacher);
        this.add(jLabel5);
        this.add(tctype);
        this.add(tcplace);
        this.add(jLabel6);
        this.add(tcno);
        this.add(jLabel4);
        this.add(jLabel3);
        this.add(jLabel1);
        this.add(jcb_cname);
        this.add(btn_exit);
        Vector v1 = cm.selectCourse();                    //调用查询所有课程名称方法
        for (int i = 0; i < v1.size();i++) {
            jcb_cname.addItem(v1.get(i).toString());      //将查询结果添加到下拉列表中
        }
        this.setVisible(true);
        this.setSize(600, 600);
    }
    JLabel jLabel1 = new JLabel();
```

```
        JLabel jLabel2 = new JLabel();
        JTextField tctime = new JTextField();
        JTextField tcno = new JTextField();
        JTextField tcteacher = new JTextField();
        JLabel jLabel3 = new JLabel();
        JLabel jLabel4 = new JLabel();
        JLabel jLabel5 = new JLabel();
        JTextField tctype = new JTextField();
        JTextField tcplace = new JTextField();
        JLabel jLabel6 = new JLabel();
        JButton btn_exit = new JButton();
        JComboBox jcb_cname = new JComboBox();
        CourseManager cm = new CourseManager();
        public void btn_exit_actionPerformed(ActionEvent e) {
            this.setVisible(false);
        }
        public void jcb_cname_itemStateChanged(ItemEvent e) {
        //调用按课程名称查询课程信息
            Course c = cm.findOneCourse(jcb_cname.getSelectedItem().toString());
            tcno.setText(c.getCno());
            tcteacher.setText(c.getCteacher());
            tcplace.setText(c.getCplace());
            tctype.setText(c.getCtype());
            tcplace.setText(c.getCplace());
            tctime.setText(c.getCtime());
        }
    }
    class Select_coursecnamePanel_jcb_cname_itemAdapter implements ItemListener {
        private Select_coursecnamePanel adaptee;
        Select_coursecnamePanel_jcb_cname_itemAdapter(Select_coursecnamePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void itemStateChanged(ItemEvent e) {
            adaptee.jcb_cname_itemStateChanged(e);
        }
    }
    class Select_coursePanel_btn_exit_actionAdapter implements ActionListener {
        private Select_coursecnamePanel adaptee;
        Select_coursePanel_btn_exit_actionAdapter(Select_coursecnamePanel adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_exit_actionPerformed(e);
        }
    }
```

17．Select_coursecteacherPanel1.java

Select_coursecteacherPanel1 类用于实现按授课教师查询课程信息界面功能，其界面如图 5-28 所示。

图 5-28　按授课教师查询课程信息界面

其代码如下：

```
package cvit.com.view;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JLabel;
import java.awt.Rectangle;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import cvit.com.manager.CourseManager;
import javax.swing.JOptionPane;
import cvit.com.model.Course;
import javax.swing.JComboBox;
import java.util.Vector;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
//根据任课老师查询
public class Select_coursecteacherPanel1 extends JPanel {
    public Select_coursecteacherPanel1() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
```

```
        this.setBackground(new Color(168, 233, 216));
        jLabel1.setText("课程编码");
        jLabel1.setBounds(new Rectangle(192, 46, 65, 15));
        tctype.setEnabled(false);
        tctype.setBounds(new Rectangle(260, 82, 77, 20));
        btn_exit.setBounds(new Rectangle(142, 178, 81, 23));
        btn_exit.setText("退出");
        btn_exit.addActionListener(new
                                   Select_coursecteacherPanel1_btn_exit_actionAdapter(this));
        tctime.setEnabled(false);
        tctime.setBounds(new Rectangle(260, 120, 77, 20));
        jLabel2.setToolTipText("");
        jLabel2.setText("授课教师");
        jLabel2.setBounds(new Rectangle(11, 41, 53, 15));
        jLabel3.setText("课程名称");
        jLabel3.setBounds(new Rectangle(18, 86, 60, 15));
        tcno.setEnabled(false);
        tcno.setBounds(new Rectangle(260, 42, 77, 20));
        jLabel4.setText("课程类别");
        jLabel4.setBounds(new Rectangle(193, 84, 62, 15));
        tcname.setEnabled(false);
        tcname.setBounds(new Rectangle(81, 83, 77, 20));
        jLabel5.setText("上课地点");
        jLabel5.setBounds(new Rectangle(19, 124, 67, 15));
        tcplace.setEnabled(false);
        tcplace.setBounds(new Rectangle(83, 123, 77, 20));
        jLabel6.setText("上课时间");
        jLabel6.setBounds(new Rectangle(194, 121, 57, 15));
        jcb_teacher.setBounds(new Rectangle(83, 36, 75, 23));
        jcb_teacher.addItemListener(new
                                    Select_coursecteacherPanel1_jcb_teacher_itemAdapter(this));
        this.add(jLabel5);
        this.add(tcname);
        this.add(jLabel3);
        this.add(jLabel2);
        this.add(jcb_teacher);
        this.add(tcplace);
        this.add(btn_exit);
        this.add(tctype);
        this.add(tctime);
        this.add(jLabel4);
        this.add(jLabel6);
        this.add(jLabel1);
        this.add(tcno);
        Vector v1 = cm.selectCourse2();                    //调用查询所有任课教师名称方法
        for (int i = 0; i < v1.size(); i++) {
            jcb_teacher.addItem(v1.get(i).toString());
```

```
        }
        this.setVisible(true);
        this.setSize(600, 600);
    }
    JLabel jLabel1 = new JLabel();
    JTextField tctype = new JTextField();
    JButton btn_exit = new JButton();
    JTextField tctime = new JTextField();
    JLabel jLabel2 = new JLabel();
    JLabel jLabel3 = new JLabel();
    JTextField tcno = new JTextField();
    JLabel jLabel4 = new JLabel();
    JTextField tcname = new JTextField();
    JLabel jLabel5 = new JLabel();
    JTextField tcplace = new JTextField();
    JLabel jLabel6 = new JLabel();
    JComboBox jcb_teacher = new JComboBox();
    CourseManager cm = new CourseManager();
    public void btn_exit_actionPerformed(ActionEvent e) {
        this.setVisible(false);
    }
    public void btn_select_actionPerformed(ActionEvent e) {
        String cteacher = jcb_teacher.getSelectedItem().toString();
        CourseManager cm = new CourseManager();
        //调用按任课教师查询课程信息方法
        Course c = cm.findCourse_teacher(cteacher);
        if (c == null) {
            JOptionPane.showMessageDialog(this, "没有该课程！", "提示信息",
                                          JOptionPane.INFORMATION_MESSAGE);
        } else {
            tcno.setText(c.getCno());
            tcname.setText(c.getCname());
            tctype.setText(c.getCtype());
            tcplace.setText(c.getCplace());
            tctime.setText(c.getCtime());
        }
    }
    public void jcb_teacher_itemStateChanged(ItemEvent e) {
        Course c = cm.findOneCourse2(jcb_teacher.getSelectedItem().toString());
        System.out.println(c.getCno());
        System.out.println(c.getCname());
        System.out.println(c.getCtype());
        System.out.println(c.getCplace());
        System.out.println(c.getCtime());
        tcno.setText(c.getCno());
        tcname.setText(c.getCname());
        tcplace.setText(c.getCplace());
```

```
            tctype.setText(c.getCtype());
            tctime.setText(c.getCtime());
        }
    }
    class Select_coursecteacherPanel1_jcb_teacher_itemAdapter implements
            ItemListener {
        private Select_coursecteacherPanel1 adaptee;
        Select_coursecteacherPanel1_jcb_teacher_itemAdapter(
                Select_coursecteacherPanel1 adaptee) {
            this.adaptee = adaptee;
        }
        public void itemStateChanged(ItemEvent e) {
            adaptee.jcb_teacher_itemStateChanged(e);
        }
    }
    class Select_coursecteacherPanel1_btn_exit_actionAdapter implements
            ActionListener {
        private Select_coursecteacherPanel1 adaptee;
        Select_coursecteacherPanel1_btn_exit_actionAdapter(
                Select_coursecteacherPanel1 adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_exit_actionPerformed(e);
        }
    }
```

18．Select_gradestudentPanel1.java

Select_gradestudentPanel1 类用于实现按学号查询并显示某一学生全部成绩界面功能，其界面如图 5-29 所示。

图 5-29　按学号查询并显示某一学生全部成绩界面

其代码如下：

```
    package cvit.com.view;
```

```
import java.awt.BorderLayout;
import cvit.com.manager.StudentManager;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.JButton;
import java.awt.Rectangle;
import javax.swing.JTextField;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JList;
import java.util.Vector;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import cvit.com.manager.GradeManager;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class Select_gradestudentPanel1 extends JPanel {
    public Select_gradestudentPanel1() {
        try {
            jbInit();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
    private void jbInit() throws Exception {
        this.setLayout(null);
        this.setBackground(new Color(168, 233, 216));
        jcb_sno.setBounds(new Rectangle(93, 20, 101, 23));
        jcb_sno.addItemListener(new
                                SelectGradestudentPanel1_jcb_sno_itemAdapter(this));
        jLabel1.setText("学号");
        jLabel1.setBounds(new Rectangle(35, 22, 55, 20));
        list.setBounds(new Rectangle(35, 68, 330, 123));
        btn_exit.setBounds(new Rectangle(237, 21, 81, 23));
        btn_exit.setText("退出");
        btn_exit.addActionListener(new
                                SelectGradestudentPanel1_btn_exit_actionAdapter(this));
        this.add(list);
        this.add(jLabel1);
        this.add(jcb_sno);
        this.add(btn_exit);
        Vector v = sm.selectStudent();                    //调用查询学号方法
        for (int i = 0; i < v.size(); i++) {
            jcb_sno.addItem(v.get(i).toString());
        }
        this.setVisible(true);
        this.setSize(600,600);
```

```
        }
        JComboBox jcb_sno = new JComboBox();
        JLabel jLabel1 = new JLabel();
        JList list = new JList();
        StudentManager sm=new StudentManager();
        JButton btn_exit = new JButton();
        public void jcb_sno_itemStateChanged(ItemEvent e) {
           String sno = (String) jcb_sno.getSelectedItem();
              GradeManager sm = new GradeManager();
                    Vector v = sm.selectGradestudent(sno);//调用某同学所有科目及成绩方法
                     list.setListData(v);
        }
        public void btn_exit_actionPerformed(ActionEvent e) {
            this.setVisible(false);
        }
    }
    class SelectGradestudentPanel1_btn_exit_actionAdapter implements ActionListener {
        private Select_gradestudentPanel1 adaptee;
        SelectGradestudentPanel1_btn_exit_actionAdapter(Select_gradestudentPanel1
                adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.btn_exit_actionPerformed(e);
        }
    }
    class SelectGradestudentPanel1_jcb_sno_itemAdapter implements ItemListener {
        private Select_gradestudentPanel1 adaptee;
        SelectGradestudentPanel1_jcb_sno_itemAdapter(Select_gradestudentPanel1 adaptee) {
            this.adaptee = adaptee;
        }
        public void itemStateChanged(ItemEvent e) {
            adaptee.jcb_sno_itemStateChanged(e);
        }
    }
```

5.5 运行与发布

5.5.1 运行

将所有的源文件保存到一个文件夹中，如 e:\StudentMS。在使用 javac 命令进行编译之前，应使用如下命令设置类的路径：

```
e:\StudentMS set class path=e:\StudentMS
```

然后利用 javac 命令对文件进行编译，使用如下命令：

```
javac  -d  e:\StudentMS  -cp  e:\StudentMS  e:\StudentMS\*.java
```

其中，-d 参数用于指明编译生成的 class 文件，放置的路径；-cp 用于指明查找当前类文件的路径，即指明类的寻找路径为包所在的路径；第三个参数中注意要指明类的完整路径。

之后，使用 java 命令执行程序：

```
java  -cp  e:\StudentMS cvit.com.view.StuMainFrame
```

程序运行。

此处需要注意的是：在使用 java 命令执行这种包文件的时候，也需要用参数-cp 指明类的寻找路径为包所在的路径，后面的参数应以“.”来间隔包类各层文件夹之间的关系。

5.5.2 发布

使用 jar.exe 将应用程序打包，把应用程序中涉及的类和图片压缩成一个 jar 文件，这样就可以发布程序了。

（1）编写一个程序文件，名称为 STUDENT.MF，代码如下：

```
Manifest –Version: 1.0
Created –By: 1.5.0_02 (Sun Microsystems Inc.)
Main−Class: StudentInfoSystem
```

MANIFEST.MF 文件保存在 e:\java\student。

（2）使用如下命令生成 jar 文件：

```
jar cfm StudentInfoSystem.jar STUDENT.MF *.class
```

（3）编写一个 bat 文件，文件名为 StudentInfoSystem.bat。

内容为：javaw –jar StudentInfoSystem.jar。

与 StudentInfoSystem.jar 保存在同一个文件夹下。

（4）运行 StudentInfoSystem.jar 文件即可。前提是计算机上安装了 Java JDK，并配置了环境变量。

5.6 本项目实现中常见问题

（1）使用 jabc 连接数据库时，一定要先引入包 class12.jar。

（2）连接数据库时，IP 地址要使用 LocalHost，而不要使用具体的 IP 地址码，如 10.1.22.35，否则会导致当系统安装到客户机器时，不指向本机的数据库，而是指向 IP 地址指向的机器的数据库。

（3）当开发查询功能模块代码时，注意不要将创建实体类对象的语句“实体类名 对象名=null;”写成“实体类名 对象名=new 实体类名（）;”；否则会导致未查找到数据时，判断失误。因为如果没有查询到记录，系统判断返回对象也不为空。

（4）数据库对象不能在类中定义，因为这样容易导致多个方法都调用数据库对象时，互相冲突。

（5）当完成添加、删除、查询、修改等功能后，注意依次关闭数据库对象。

5.7 项目技术支持

5.7.1 什么是JDBC

JDBC，全称为 JavaDataBaseConnectivitystandard，是一个面向对象的应用程序接口（API），通过它可访问各类关系数据库。JDBC 也是 Java 核心类库的一部分。

JDBC 的最大特点是独立于具体的关系数据库。JDBCAPI 中定义了一些 Java 类分别用来表示与数据库的连接（connections），SQL 语句（SQLstatements）、结果集（resultsets）及其他的数据库对象，使得 Java 程序能方便地与数据库交互并处理所得的结果。使用 JDBC，所有 Java 程序（包括 Java applications,applets 和 servlet）都能通过 SQL 语句或存储在数据库中的过程（stored procedures）来存取数据库。

JDBC 由一组用 Java 编程语言编写的类和接口组成，主要放在 java.sql.*包中。JDBC 为工具/数据库开发人员提供了一个标准的 API，使他们能够用纯 JavaAPI 来编写数据库应用程序。

有了 JDBC，向各种关系数据库发送 SQL 语句就是一件很容易的事。换言之，有了 JDBCAPI，就不必为访问 Sybase 数据库专门写一个程序，为访问 Oracle 数据库又专门写一个程序，为访问 Informix 数据库又写另一个程序，等等。您只需用 JDBCAPI 写一个程序就够了，它可向相应数据库发送 SQL 语句。而且，使用 Java 编程语言编写的应用程序，无须去忧虑要为不同的平台编写不同的应用程序。将 Java 和 JDBC 结合起来将使程序员只须写一遍程序就可让它在任何平台上运行。

5.7.2 两层模型和三层模型

JDBCAPI 既支持数据库访问的两层模型，同时也支持三层模型。

在两层模型中，Javaapplet 或应用程序将直接与数据库进行对话。这将需要一个 JDBC 驱动程序来与所访问的特定数据库管理系统进行通信。用户的 SQL 语句被送往数据库中，而其结果将被送回给用户。数据库可以位于另一台计算机上，用户通过网络连接到上面。这就叫做客户机/服务器配置，其中用户的计算机为客户机，提供数据库的计算机为服务器。网络可以是 Intranet（它可将公司职员连接起来），也可以是 Internet。

在三层模型中，命令先是被发送到服务的“中间层”，然后由它将 SQL 语句发送给数据库。数据库对 SQL 语句进行处理并将结果送回到中间层，中间层再将结果送回给用户。MIS 主管们都发现三层模型很吸引人，因为可用中间层来控制对公司数据的访问和可做的更新的种类。中间层的另一个好处是，用户可以利用易于使用的高级 API，而中间层将把它转换为相应的低级调用。许多情况下三层结构可提供一些性能上的好处。

5.7.3 解析JDBC

使用 JDBC 连接数据，主要使用的接口或类有以下 4 个。

1．Connection

Connection 对象代表与数据库的连接。连接过程包括所执行的 SQL 语句和在该连接上所返回的结果。一个应用程序可与单个数据库有一个或多个连接，或者可与许多数据库有连接。

另外，需要注意的是：默认情况下，Connection 对象处于自动提交模式下，这意味着它在执行每个语句后都会自动提交更改。如果禁用自动提交模式，为了提交更改，必须显式调用 commit 方法；否则无法保存数据库更改。

2．Statement

用于执行静态 SQL 语句并返回它所生成结果的对象，它提供了执行语句和获取结果的基本方法。

（1）创建 Statement 对象。建立了到特定数据库的连接之后，就可用该连接发送 SQL 语句。Statement 对象用 Connection 的方法 createStatement 创建，如下列代码段中所示：

```
Connectioncon=DriverManager.getConnection(url,"scott","tiger");
Statementstmt=con.createStatement();
```

为了执行 Statement 对象，被发送到数据库的 SQL 语句将被作为参数提供给 Statement 的方法：

```
ResultSetrs=stmt.executeQuery("SELECT*FROMStudentInfo");
```

（2）使用 Statement 对象执行语句。Statement 接口提供了 3 种执行 SQL 语句的方法：executeQuery、executeUpdate 和 execute。使用哪一个方法由 SQL 语句所产生的内容决定。

方法 executeQuery 用于产生单个结果集的语句，如 SELECT 语句。

方法 executeUpdate 用于执行 INSERT、UPDATE 或 DELETE 语句以及 SQLDDL（数据定义语言）语句，如 CREATETABLE 和 DROPTABLE。INSERT、UPDATE 或 DELETE 语句的效果是修改表中零行或多行中的一列或多列。executeUpdate 的返回值是一个整数，指示受影响的行数（即更新计数）。对于 CREATETABLE 或 DROPTABLE 等不操作行的语句，executeUpdate 的返回值总为零。

方法 execute 用于执行返回多个结果集、多个更新计数或二者组合的语句。因为多数程序员不需要该高级功能，所以本概述后面将在单独一节中对其进行介绍。

执行语句的所有方法都将关闭所调用的 Statement 对象的当前打开结果集（如果存在）。这意味着在重新执行 Statement 对象之前，需要完成对当前 ResultSet 对象的处理。

应注意，继承了 Statement 接口中所有方法的 PreparedStatement 接口都有自己的 executeQuery、executeUpdate 和 execute 方法。Statement 对象本身不包含 SQL 语句，因而必须给 Statement.execute 方法提供 SQL 语句作为参数。PreparedStatement 对象并不将 SQL 语句作为参数提供给这些方法，因为它们已经包含预编译 SQL 语句。CallableStatement 对象继承这些方法的 PreparedStatement 形式。对于这些方法的 PreparedStatement 或 CallableStatement 版本，使用查询参数将抛出 SQLException。

（3）语句完成。当连接处于自动提交模式时，其中所执行的语句在完成时将自动提交或还原。语句在已执行且所有结果返回时，即认为已完成。对于返回一个结果集的 executeQuery 方法，在检索完 ResultSet 对象的所有行时该语句完成。对于方法 executeUpdate，当它执行时语句即完成。但在少数调用方法 execute 的情况中，在检索所有结果集或它生成的更新计数之后语句才完成。

有些 DBMS 将已存储过程中的每条语句视为独立的语句；而另外一些则将整个过程视为一个复合语句。在启用自动提交时，这种差别就变得非常重要，因为它影响什么时候调用 commit 方法。在前一种情况中，每条语句单独提交；在后一种情况中，所有语句同

时提交。

（4）关闭 Statement 对象。Statement 对象将由 Java 垃圾收集程序自动关闭。而作为一种好的编程风格，应在不需要 Statement 对象时显式地关闭它们。这将立即释放 DBMS 资源，有助于避免潜在的内存问题。

3．ResultSet

ResultSet 表示数据库结果集的数据表，通常通过执行查询数据库的语句生成。

ResultSet 对象具有指向其当前数据行的指针。最初，指针被置于第一行之前。next 方法将指针移动到下一行；因为该方法在 ResultSet 对象中没有下一行时返回 false，所以可以在 while 循环中使用它来迭代结果集。

默认的 ResultSet 对象不可更新，仅有一个向前移动的指针。因此，只能迭代它一次，并且只能按从第一行到最后一行的顺序进行。可以生成可滚动和（或）可更新的 ResultSet 对象。以下代码片段（其中 con 为有效的 Connection 对象）演示了如何生成可滚动且不受其他更新影响的、可更新的结果集。请参阅 ResultSet 字段以了解其他选项。

```
Statementstmt=con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_
UPDATABLE);
```

4．PreparedStatement

PreparedStatement 表示预编译的 SQL 语句的对象。

SQL 语句被预编译并且存储在 PreparedStatement 对象中。然后可以使用此对象高效地多次执行该语句。

PreparedStatement 实例包含已编译的 SQL 语句。这就是使语句“准备好”。包含于 PreparedStatement 对象中的 SQL 语句可具有一个或多个 IN 参数。IN 参数的值在 SQL 语句创建时未被指定。相反的，该语句为每个 IN 参数保留一个问号（“？”）作为占位符。每个问号的值必须在该语句执行之前，通过适当的 setXXX 方法来提供。

PreparedStatement 接口继承 Statement，但与 Statement 在以下两方面有所不同。

第一，由于 PreparedStatement 对象已预编译过，所以其执行速度要快于 Statement 对象。因此，多次执行的 SQL 语句经常创建为 PreparedStatement 对象，以提高效率。

第二，作为 Statement 的子类，PreparedStatement 继承了 Statement 的所有功能。另外它还添加了一整套方法，用于设置发送给数据库以取代 IN 参数占位符的值。同时，3 种方法 execute、executeQuery 和 executeUpdate 已被更改以使之不再需要参数。这些方法的 Statement 形式（接受 SQL 语句参数的形式）不应该用于 PreparedStatement 对象。

（1）创建 PreparedStatement 对象。以下的代码段（其中 con 是 Connection 对象）创建包含带两个 IN 参数占位符的 SQL 语句的 PreparedStatement 对象。

```
PreparedStatementpstmt=con.prepareStatement("UPDATEtable4SETm=?WHEREx=?");
```

pstmt 对象包含语句"UPDATEtable4SETm=?WHEREx=?"，它已发送给 DBMS，并为执行做好了准备。

（2）传递 IN 参数。在执行 PreparedStatement 对象之前，必须设置每个?参数的值。这可通过调用 setXXX 方法来完成，其中 XXX 是与该参数相应的类型。例如，如果参数具有 Java

类型 long，则使用的方法就是 setLong。setXXX 方法的第一个参数是要设置的参数的序数位置，第二个参数是设置给该参数的值。例如，以下代码将第一个参数设为 123456789，第二个参数设为 100000000：

```
pstmt.setLong(1,123456789);
pstmt.setLong(2,100000000);
```

一旦设置了给定语句的参数值，就可用它多次执行该语句，直到调用 clearParameters 方法清除它为止。在连接的默认模式下（启用自动提交），当语句完成时将自动提交或还原该语句。

如果基本数据库和驱动程序在语句提交之后仍保持这些语句的打开状态，则同一个 PreparedStatement 可执行多次。如果这一点不成立，那么试图通过使用 PreparedStatement 对象代替 Statement 对象来提高性能是没有意义的。

利用 pstmt（前面创建的 PreparedStatement 对象），以下代码例示了如何设置两个参数占位符的值并执行 pstmt 10 次。如上所述，为做到这一点，数据库不能关闭 pstmt。在该示例中，第一个参数被设置为"Hi"并保持为常数。在 for 循环中，每次都将第二个参数设置为不同的值：从 0 开始，到 9 结束。

```
pstmt.setString(1,"Hi");
for(int i=0;i<10;i++){
pstmt.setInt(2,i);
introwCount=pstmt.executeUpdate();
}
```

（3）IN 参数中数据类型的一致性。setXXX 方法中的 XXX 是 Java 类型。它是一种隐含的 JDBC 类型（一般 SQL 类型），因为驱动程序将把 Java 类型映射为相应的 JDBC 类型（遵循该 JDBCGuide 中 §8.6.2“映射 Java 和 JDBC 类型”表中所指定的映射），并将该 JDBC 类型发送给数据库。例如，以下代码段将 PreparedStatement 对象 pstmt 的第二个参数设置为 44，Java 类型为 short。

```
pstmt.setShort(2,44);
```

驱动程序将 44 作为 JDBCSMALLINT 发送给数据库，它是 Javashort 类型的标准映射。

程序员的责任是确保将每个 IN 参数的 Java 类型映射为与数据库所需的 JDBC 数据类型兼容的 JDBC 类型。不妨考虑数据库需要 JDBCSMALLINT 的情况。如果使用方法 setByte，则驱动程序将 JDBCTINYINT 发送给数据库，这是可行的，因为许多数据库可从一种相关的类型转换为另一种类型，并且通常 TINYINT 可用于 SMALLINT 适用的任何地方。

5.7.4 JDBC如何连接数据库

JDBC 连接数据库，可以分成 5 步，如图 5-30 所示。

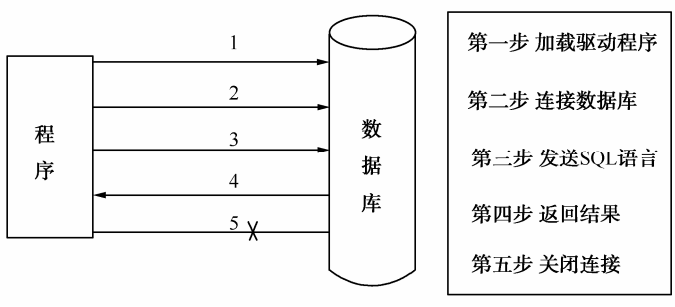

图 5-30　JDBC 连接数据库步骤

下面以连接 oracle 数据库为例，说明连接数据库的过程。

准备步骤：导入 oracle 数据库驱动程序。

```
oracle\ora92\jdbc\lib\classes12.zip
```

定义访问数据库用到的类：

```
Private Stringdriver="oracle.jdbc.driver.OracleDriver";
private Stringurl="jdbc:oracle:thin:@localhost:1521:ora";
private Connectionconn=null;                    //连接数据库对象
private Statementstmt=null;                     //能够跑 sql 语句对象
private ResultSetrs=null;                       //执行 sql 语句的返回结果
private Stringuser="scott";
private Stringpwd="tiger";
```

（1）加载驱动程序：

```
try{
    Class.forName(driver);
    } catch (ClassNotFoundException ex) {
     System.out.println("加载驱动程序有错误,驱动程序类不存在");
}
```

（2）连接数据库：

```
try{
      conn = DriverManager.getConnection(url, user, pwd);
}catch(SQLExceptionex1){
    System.out.print("取得连接的时候有错误,请核对用户名和密码");
}
```

（3）创建 stmt 对象：

```
stmt=conn.createStame();
```

（4）执行 SQL 语句：

```
//向表中添加数据
String sql="insert into student values('Jack','070111',20)";
int i=stmt.executeUpdate(sql);
//删除数据
```

```
String sql="delete from student where no='070111'";
int i= stmt.executeUpdate(sql);
//修改数据
String sql="update student set name='张三峰'whereno='070113'";
int i= stmt.executeUpdate(sql);
//浏览数据
String sql="select*from student";
rs=stmt.executeQuery(sql);
//遍历 student 表的数据
while(rs.next()){
//处理一行记录
System.out.print(rs.getString(1)+"");
System.out.print(rs.getString(2)+"");
System.out.println(rs.getInt(3));
}
```

（5）关闭数据库：

```
//关闭数据库
stmt.close();
conn.close();
```

5.8 实训

1．题目

完善学生信息管理系统，包括对教师信息的管理及用户权限的管理。

2．要求

（1）实现对教师信息的管理，可以增、删、改、查教师信息。

（2）实现一门课程可以由多名教师承担，学生选课时，不仅可以选择某门课程，还可以选择授课教师。

（3）添加登录模块，实现身份验证、用户权限管理：

① 以学生身份登录系统时，可以使用系统选择课程、查询成绩、查询学生信息等功能。

② 以教师身份登录系统时，可以使用录入成绩和修改成绩功能。

③ 以教务科人员身份登录系统时，可以增、删、改、查课程信息、学生信息、教师信息及查看学生成绩。

（4）界面要美观大方，布局合理。

参 考 文 献

1 （美）埃史尔.（译）陈昊鹏. Java 编程思想（第 4 版）（thinking in java）. 北京：机械工业出版社，2007

2 耿祥义. Java 基础教程. 北京：清华大学出版社，2004

3 （美）霍顿. Java2 入门经典（JDK5）. 北京：机械工业出版社，2006

4 孙卫琴. Java 面向对象编程. 北京：电子工业出版社，2006

5 李钟尉，马文强，陈丹丹. Java 从入门到精通. 北京：清华大学出版社，2008

6 刘钊，边小勇. Java 程序设计基础. 北京：清华大学出版社，2007

7 张广彬，孟红蕊，张永宝. Java 课程设计案例精编. 北京：清华大学出版社，2007

反侵权盗版声明